AFIT/GSS/LAS/93D-1

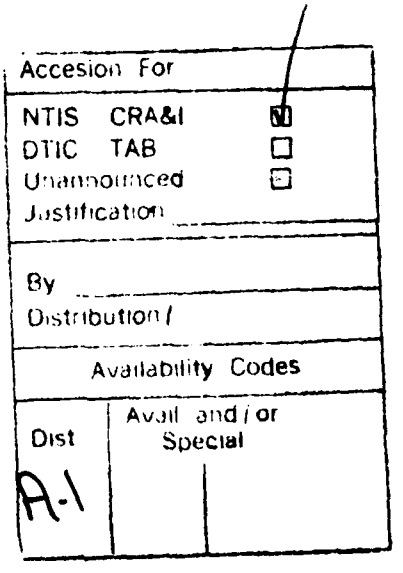
GUIDELINES FOR ENSURING SOFTWARE SUPPORTABILITY IN SYSTEMS DEVELOPED UNDER THE INTEGRATED WEAPON SYSTEM MANAGEMENT CONCEPT

THESIS

Forrest F. Butts, III, Captain, USAF
Anthony C. Johndro, Captain, USAF

AFIT/GSS/LAS/93D-1

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.



AFIT/GSS/LAS/93D-1

GUIDELINES FOR ENSURING SOFTWARE SUPPORTABILITY IN SYSTEMS DEVELOPED UNDER THE INTEGRATED WEAPON SYSTEM MANAGEMENT CONCEPT

THESIS

Presented to the Faculty of the

Graduate School of Logistics and Acquisition Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Software Systems Management

Anthony C. Johndro, B.S

Captain, USAF

Forrest F. Butts, III, B.S.

Captain, USAF

December 1993

Table of Contents

List of Figures

List of Tables

Abstract

This thesis studied the software maintenance planning and practices of the pilot Integrated Weapon System Management (IWSM) programs. Before IWSM, a System Program Office (SPO) acquired an Air Force weapon system then passed it to an Air Logistics Center (ALC) for follow-on support. The ALC was forced to maintain the software despite its questionable maintainability. The SPO de-emphasized maintainability because maintenance was an ALC responsibility and building maintainable software increased development costs and lengthened schedules. The IWSM philosophy closes the gap between development and maintenance. A System Program Director (SPD) ,who oversees both the system development and maintenance, has an inherent interest in developing maintainable software because he or she is now also responsible for supporting it. This research was accomplished through a literature review of current maintainability plans and practices, followed by a survey of the pilot IWSM programs. This information was combined to form draft guidelines for ensuring software maintainability. The draft guidelines were then validated by experts in the field of software maintenance who offered opinions and recommendations on the guidelines. The guidelines stress both up from planning and techniques for improving maintainability during software development. The final guidelines are presented in Appendix B.

GUIDELINES FOR ENSURING SOFTWARE SUPPORTABILITY IN SYSTEMS DEVELOPED UNDER THE INTEGRATED WEAPON SYSTEM MANAGEMENT CONCEPT

I. Introduction

1.1 Current State of Software Maintainability

In 1992, companies in the United States spent $30 billion maintaining software. This number represents anywhere from 60 to 80 percent of each company's software budget. The estimate is that by 1995, the number will grow to 90 percent (Sherer, 1992:70). The following chart shows past and predicted Department of Defense expenditures for embedded software.

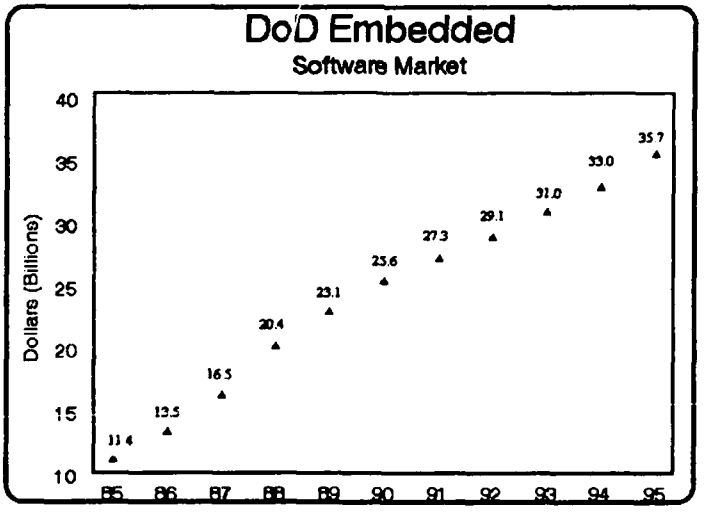


Figure 1 DOD Embedded Software Market

Applying the 60 to 80 percent to the 1992 figure ($29.1 billion), the DoD spent anywhere from $17.46 billion to $23.28 billion supporting developed or acquired software. By 1995, the estimates for DoD software support could reach $32.13 billion (MCCR,1990:2-4). Some companies must increase their maintenance staffs by 15 percent a year just to keep up with the growing demands for system changes (Sherer,

1992:70). The majority of these changes involve user enhancements to the system once it is operational (Edelstein and Mamone, 1992:82). The cause of this high cost of software support lies in the differences between software developers and maintainers. Software developers are only concerned with building and fielding a new system, with no regard for future changes to the software, while maintainers want a system that can be easily and effectively modified. Once the maintenance organization has taken responsibility for the system, then the maintainers must live with whatever software has been provided by the developers. If the maintenance organization is separate from the development organization, then maintainers have had very little interaction with the developers (Arnold, 1987:24). Also, the software developer is also more concerned with fulfilling contractual obligations (actual code and its documentation) than with providing maintainable software (Arnold, 1987:27). Software maintainers also want a system which meets the user's performance requirements, but they (maintainers) need a system which can be easily modified in order to provide effective support.

David A. Sunday of the Northrop Corporation defines software maintainability as " a software characteristic which reflects the degree of effort required to accomplish the following tasks:

1. Correction of errors.
2. Addition of features.
3. Deletion of capabilities.
4. Adaptation/Modification" (Sunday, 1989:50).

So maintainability should be thought of as a desired characteristic of software and not just another phase of the software life cycle. While the major goal of the software developers is to get the software completed on time and within budget, they should also have a goal of making the software maintainable, with particular emphasis on techniques which make enhancements to the software easier to implement (Sunday, 1989:50). The next two sections discuss past and present Air Force acquisition policy and its effect on maintainability.

1.2 Past Air Force Acquisition Policy

Before Integrated Weapon System Management (IWSM) and the creation of Air Force Materiel Command (AFMC), a weapon system was developed or acquired by a system program office (SPO), then transitioned to an Air Logistics Center (ALC) for follow on support. The goal of the SPO was to get the software developed on time and within budget. Once the software was passed to the ALC for follow on support, the SPO's job was over, and the ALC was forced to maintain or contract out maintenance for the developed software no matter what the software quality was. In short, the SPO was not concerned with maintainability because:

1. Maintenance was the responsibility of the ALC
2. To the system acquirer and developer there was a perception that building maintainability into the code would increase development costs and schedule.

The Air Force, in response to the increased support costs, searched for a new way to acquire software.

1.3 Acquisition Policy with IWSM

To close the gap between development and maintenance, the Air Force created the IWSM concept. With IWSM, the single manager, called the System Program Director (SPD) oversees both the development/acquisition and the system maintenance, as well as the transition between the two phases. The SPD now has a inherent interest in developing maintainable software because he or she is now ultimately responsible for supporting the software he or she developed. The ultimate goal of IWSM is to provide a better product to the customer. With the majority (up to 80%) of a software system life cycle's time, effort and budget going to support of the software after development ends, the SPD should pay particular attention to techniques which would improve the maintainability of the software (Edelstein and Mamone, 1992:82).

1.4 Benefits of Maintainable Software

Any improvements in maintainability will reduce both the effort, through improved productivity because the software is more easily corrected, and cost of supporting the software because fewer problems can be repaired faster. Maintainable software provides the customer with a higher quality product because there are fewer problems in the operational system. Wilma Osborne of the Institute for Computer Sciences and Technology relates, "Software maintenance is an integral part of the software development life cycle and should not be considered a separate entity with respect to staffing, management support, and other resources" (Osborne, 1987:13).

1.5 Maintainability Failures and Successes

Thomas M. Pigoski and Craig A. Cowden related both the benefits of having, and the consequences of not having, maintainability in the developed software in the article "Software Transition: Experience and Lessons Learned" at the 1992 IEEE Conference on Software Maintenance. For a system accepted from a Navy Research and Development Laboratory, there was no involvement by the maintenance organization during development, and, as a result, the system had major problems when deployed. There was no maintenance plan, no attendance at design reviews, no coordination on configuration management activities, and no participation in the testing. In fact, the system was to be fielded before the maintenance contract was even awarded. When the system was finally fielded , the users were dissatisfied with what they received. The developing organization tried to repair the problems, without the benefit of any configuration management procedures or documentation, and was unsuccessful. When a maintenance contract was finally awarded, the maintaining organization was not immediately ready to support the system. The maintainers spent weeks with the developer learning the code because the developers failed to develop a plan for training the maintainers. Then the maintainers had to establish a set of configuration management procedures and bring out-of-date designs up to date. The bottom line is that each of these activities cost money, money which could have been saved by involving the maintainers in the system's development.

The same authors were also involved with another system accepted from a DOD organization, except this time the maintainers were involved with the system development. The maintainers participated in the reviews, reviewed all documentation, received training from the developers, and even participated in testing. The system was completed on time and within budget. The user was extremely satisfied with the product, and the maintenance organization was prepared to immediately support the system, which was released just prior to the publication of the article (Pigoski and Cowden, 1992:295-297).

Another success story was presented by Rodger L Fritz and Fred Shocket at the 1988 IEEE Conference on Software Maintenance. They reported on the software maintenance success of their Light Airborne Multipurpose System (LAMPS) built by IBM for the U.S. Navy. The developers used many of the same techniques as the company in the previous example, such as creation of a software support plan up front, the selection of the maintainers early in the development, and participation in reviews. An additional technique was the participation in inspections by the support personnel. All high-level and detailed designs were inspected. Code inspections were used selectively based on the criticality of the module. The system was finished under schedule and under budget. Maintenance costs have been kept low because these various techniques resulted in relatively defect free software and minimal problems in enhancing the software. These final two examples show the benefits of maintainable software: a better product with fewer errors which can be easily enhanced. This better product also saves money because errors are caught before the system is operational, and the effort and cost to enhance the software is reduced (Fritz and Shocket, 1988:165-167).

These failures and successes have common features in that planning for maintainability and the implementation of supportability techniques affect the maintainability of a software system.

1.6 Problem Statement

How should the system support manager (SSM) plan for supportability and involve software maintainers during development to improve the maintainability of weapon system and Command Control Communications, Computers, and Intelligence (C4I) software developed under the IWSM concept?

1.7 Research Objectives

The research objectives are to determine how the software support managers can plan for Post Deployment Software Support (PDSS) and how the software maintainers can participate during the development/acquisition of weapon system software to improve the supportability of the software system.

Several questions form the framework of our investigation:

1. What maintainability planning have the program management personnel done for software systems currently under development?
2. Are the software maintainers participating in the development of the software to improve maintainability?
3. For systems already in the support phase:
   A. Was maintainability planning done by the software support management personnel during development?
   B. Did the software maintainers participate in the development of the software to improve maintainability?
   C. How satisfied are the software maintainers with the system they must maintain?
   D. What do the software maintainers wish could have been done differently during software development to make the software maintenance easier?
   E. Is there any correlation between early and effective maintainability planning and implementation and the satisfaction of the software maintainer?
4. How do the current software support plans compare with past plans?
5. What planning techniques do the experts recommend for improving maintainability?
6. How do these techniques compare with techniques used in current and past weapon system software development?

7. Can past maintainability plans be combined with current efforts and expert opinion to provide a set of guidelines for future maintainability efforts?

1.8 Scope and Limitations

The research is limited to an Air Force population of 22 weapon systems and C4I systems currently under IWSM. The SPD, who is responsible for the entire life cycle cost and effort, can more readily apply the improved maintainability techniques than the separate development and maintenance managers can under the non-IWSM process. The maintainability techniques are also geared towards the acquisition of software (government procured custom software), as opposed to development of software (government developed software).

1.9 Significance of Research

The ultimate goal of IWSM is to provide a better product to the customer. The techniques for improving maintainability stress requirements validity, clear design, understandable code, effective testing, clear configuration management plans, and involvement of quality assurance teams. While these are all positive effects of improved maintainability, the real benefits of a maintainable system are a reduced life cycle cost and a better operational system. It is cheaper to fix problems before the system becomes operational. A system developed for maintainability has fewer problems, which can be repaired faster, and the software can also be changed to meet new requirements more effectively. "A lack of attention to software maintainability during the requirements specification, design, and coding phases generally leads to excessive software maintenance costs" (Osborne, 1987:21).

1.10 Definitions of Terms

C4I - Command, Control, Communications, Computers, and Intelligence

Development Systems Manager (DSM) - The individual responsible for the initial development of a weapon system. Reports to the System Program Director.

IWSM - Integrated Weapon System Management

System Lifecycle - The life of a weapon system from initial concept to retirement.

Software Support - Perfective, corrective, and adaptive changes to software.

System Program Director (SPD) - The single manager for acquisition and support of a weapon system.

System Support Manager (SSM) - The individual responsible for the support of a weapon system. Reports to the System Program Director.

1.11 Overview

The next chapter contains a literature review of the IWSM concept and techniques for improving maintainability in the development of software systems. Chapter 3 contains the methodology to answer the questions posed in section 1.7. Chapter 4 contains the analysis of data collected and a description of the effort to construct guidelines for ensuring supportability. Finally, chapter 5 contains the conclusions reached from the analysis and recommendations for improvement and follow on research.

II. Literature Review

2.1 Introduction

This chapter examines the current literature available on building maintainability into software systems and the growing management philosophy of Integrated Weapon System Management (IWSM). The review is divided into five sections: planning for Post Deployment Software Support (PDSS), PDSS acquisition requirements, ensuring software supportability and quality, supportability techniques in the development cycle, and, finally, background on IWSM. The first two sections assume the development and maintenance contracts have not been awarded yet.

The PDSS process starts during the development of a software system. Several organizations share responsibility for PDSS *planning, requirements, preparation,* and *performance*. Briefly, the user organization, or customer, identifies the need for the weapon system (which may require software). They also plan for PDSS with a system maintenance concept, or, a suggested approach to perform system maintenance. A Program Management Office (PMO) or System Program Office (SPO) (also the procuring agency) is given the charge of planning, budgeting, managing and acquiring the weapon system. The PMO/SPO assists the user in planning and preparing for future PDSS. The PMO/SPO typically contracts with a development organization to develop the software system. The development organization, or developer, may be a civilian company or a military organization with the resources and expertise to design and develop the software system. The development organization produces the software system in accordance with the user's specifications. Also, the developer considers plans, designs, and implementation of software supportability within the software system. In conjunction with these organizations, the maintenance organization, or maintainer (also Software Support Agency (SSA)) oversees the software's development to ensure supportability issues are addressed. During development, the initial maintainer is typically a military organization. However, like the development, actual software support may be performed by a civilian company. Together, these four groups form the weapon system acquisition and support team.

According to MIL-HDBK-347, planning is one of the most important activities to reduce the cost and risk of software maintenance. The earlier the planning begins, the better the support will be. The plan outlines the procedures for maintaining the system. The plan includes:

- Organization
- Hardware
- Software
- Personnel requirements
- Facilities requirements.

The plan also covers the program office's relationship with the user and maintainer, how requirements changes are incorporated into the system, along with quality assurance and testing of each changed version (Osborne, 1987:21, Arnold, 1987:30, MIL-HDBK-347,1990:18).

Before the system is acquired, a defined maintenance concept of operations gives the procuring agency a basis for evaluating the maintainability of the system. "The key idea is that the concept of operations must drive the maintenance of the delivered system, not the system driving how maintenance is performed" (Arnold, 1987:30). A good transition plan eases the strain of moving from a development to a support environment. Supportability techniques, such as structured programming and coding standards, reduce software complexity, making it easier for a programmer to make changes. Configuration management ensures supportability through an effective change control process. Effective testing and debugging not only reduce the number of errors in a software system, but also determine the effectiveness of the error correction process. Quality assurance improves maintainability through the effective use of reviews and audits, along with design and code walkthroughs, help to ensure correct specifications and implementations. The first step is to plan for resources.

2.2 Plan for PDSS

According to MIL-HDBK-347, the PDSS concept is the basis for the whole PDSS plan. The concept describes how and to what extent the software will be maintained. SSA resource requirements planning identifies the resources required to implement the PDSS concept (MIL-HDBK-347, 1990:18-19). The procedures also define facilities, equipment and personnel requirements, along with supportability characteristics of the software, to include the means of measuring those characteristics (Sunday, 1989:51). If at all possible, the maintenance organization should be identified or the contract awarded when the plan is formed. If they are not identified, then the actual functions expected of an SSA should be laid out. The user's organization interface with the SSA should also be identified. The SSA's functions group into the following categories: management and administration, software engineering and test, configuration management, software generation and distribution, technical documentation, deployment and installation, and quality assurance (Vollman, 1990:193).

2.2.1 Management and Administration. A management structure coordinates and controls not only the SSA activities, but also the various organizations involved. Memorandums of Agreement (MOA) between program management, the user, and the maintainers on each organization's responsibilities should be drawn up. The MOA should require the maintaining organization to review all documentation, participate in formal walkthroughs, and participate in reviews and testing during software development. The user and maintenance organization relationship needs to be established early, because they have to work together with the developed software for 10 to 15 years after system completion. The development and support organizations need to establish a good relationship because they are both working towards the same goal, a high quality system. The PMO is the hub or connection between these three entities. The PMO keeps the lines of communication flowing, and ensures that the right people from the right organizations can meet whenever needed. A high quality, supportable system requires the effort of all four organizations. During the planning phase, personnel requirements should be identified for both the complete and minimum system transition staffs. Determining SSA facility requirements along with the system hardware and software suites needs are part of the planning.

2.2.2 Software Engineering and Test.

This function ensures that the software system is modifiable and testable. Robert Arnold states that even though the developer will spend a great amount of time developing a set of test data, it may not be used by the maintainers (Arnold, 1987: 27). Some reasons for this are: the test data was not a deliverable item under the development contract, the maintainers may not have the proper tools for testing software, and the data itself may be hard or impossible to understand. To ensure the test data can be reused make sure it is a deliverable and, if possible, have the test data jointly developed by the developers and maintainers. If this is not possible, the maintenance organization should review all test procedures and attend the Test Readiness Review to ensure they can use the test data. The test procedures should describe the tools needed to maintain or test the system. The following tools can be helpful:

- Cross Reference/Browsers - These types of tools show where variable names are used throughout the system so they can be changed consistently.
- Performance Analyzers - These tools are effective in determining which modules are slowing down the system at execution time.
- Code Auditors - These provide automated assistance in checking for standards confirmation.
- Comparator - This tool compares files to determine differences in files from one version to another.
- Configuration Management/Version Control - These protect the baseline versions from becoming inadvertently changed.
- Requirements Tracer - This tool traces requirement changes back through the specifications.
- Ripple Effect Detector - This tool reveals all connections to a particular change to avoid unwanted side effects (Glass, 1992:197-204).

2.2.3 Configuration Management.

Configuration management identifies how the software and its documentation will be controlled. An effective change control process improves the supportability of the software. Procedures should determine how the user reports problems, how the change requests are approved and who approves the requests. Then the procedures should identify how changes are scheduled, implemented and tested. Finally, the procedures identify how the new version will be released (Osborne, 1987:21).

Configuration management identifies which documents will be required by the maintenance organization. Besides the deliverable Data Item Descriptors, the supporting organization may require data dic-

tionaries, program designs, and documentation templates. All documentation should be delivered in computer readable format as well as hard copy (Arnold, 1987:30).

2.2.4 Software Generation and Distribution. This function is concerned with how new software versions, their accompanying documentation, and user training will be distributed to the user's sites. This includes media duplication, packaging, and a physical audit of the packages (Vollman, 1990:195).

2.2.5 Technical Documentation. Technical documentation requires that a library be created to maintain the volumes of documentation and magnetic media which accompany DoD software systems. A technical library stores the source and object code for every release, all test cases, all documentation, and any tools (Humphrey, 1990: 132-133).

2.2.6 Deployment and Installation. If the SSA will be responsible for installing new versions at the user's site then personnel and procedures should be identified for installation and testing plus providing any other technical support or training required by the user (Vollman, 1990:197).

2.2.7 Quality Assurance. The plan should identify if this system is to have its own Software Quality Assurance (SQA) or if will it be part of an organization-wide SQA program. Personnel and procedures should be drawn up in either case (Vollman, 1992:193-196). The main function of SQA is the enforcement of standards. The development and support organization may have two different sets of standards, and the maintainers may not be willing or able to conform to a new set of standards. A common set of standards, to include coding practices, should be identified and documented in this section which will used by both the development and support organization. Each organization, including the program office, will be responsible for enforcing the standards (Vollman, 1990:193-197).

After determining what is required to support the system, the requirements from sections 2.2.1 through 2.2.7 can be passed on to the development contractor.

2.3 PDSS Acquisition Requirements

MIL-HDBK-347 describes PDSS acquisition requirements as contractual requirements imposed upon the software development contractor which facilitate maintainability. These requirements consist of options which could reduce life cycle costs, ease the transition to maintenance, or improve maintainability or quality. When these requirements are included in the contract, life cycle costs and risks in maintenance are reduced. Many of these acquisition requirements can be taken directly from the PDSS plan. The acquisition requirements are broken into software environment, technical data, software quality, and transition (MIL-HDBK-347, 1990: 19-20).

2.3.1 Software Environment Requirements. Software environment requirements (which include hardware) can be specified in the contract to:

- Ensure the development and support environment (including automated tools, configuration management documentation, and network protocol) are the same, or as close as possible.
- Use existing government resources as components of the software maintenance facility.
- Provide backup systems at the software support facility.
- Limit the number of different environments and development
- Implement a consolidated support concept (MIL-HDBK-347, 1990:20).

2.3.2 Technical Data Requirements. Technical data (i.e., documentation) should be identified with clear and precise requirements. While development costs can be reduced by limiting the documentation requirements, not having enough documentation can increase the follow on support costs (MIL-HDBK-347, 1990:20-21).

2.3.3 Software Quality Requirements. Software quality is also of importance to the SSA because a higher quality product is easier to maintain. The maintenance organization should be involved to improve quality, such as documentation and reviews, along with inspections and test participation. Also included should be any joint SQA ventures, such as audits between the two organizations. The program office should work with the development and support organizations to identify and establish quality requirements, a quality evaluation process, and acceptance criteria (MIL-HDBK-347, 1990:21).

2.3.4 Transition Requirements. The program office, along with the development and support organization, should establish specific transition requirements. Such requirements can include:

1. Ensure the SSA is staffed, trained and ready to support the system.
2. Install and test the hardware.
3. Install and test the software suite, including tools and databases.
4. Ensure all configuration management functions including updated documentation and records are in place.
5. Secure any licensing agreements or warranties.
6. Ensure the SSA is ready to support the user with no interruption in service (Vollman, 1990:191).

2.4 Ensure Software Supportability and Quality

Software quality and supportability go hand-in-hand during development. Any methods to improve quality improves supportability and any attempts to improve supportability result in a higher quality product. Software supportability and quality cannot be ensured by contract specifications. The SSA must actively participate in evaluating software maintainability, authenticating specifications, verifying requirements, and evaluating software quality plans and activities (MIL-HDBK-347, 1990:21-22). Software maintainability and quality can be improved by using such techniques as high level languages, structured programming, design reviews, coding standards, a maintainability checklist, code walkthroughs and inspections, meaningful comments within source code, and testing.

2.4.1 High Order Languages. Using a high order language improves software quality and supportability. The Air Force mandated the use of the programming language Ada in 1990. The language has a number of built-in facilities which simplify implementation details, thereby reducing the intellectual effort and the errors which accompany this effort (Guidelines, 1992:5-4). Ada is designed to make use of the principles of information hiding and data abstractions. James Hager of HRB systems recommends using information hiding and abstraction of interfaces to ensure quality and maintainability (Hager, 1989:272). Ada programs interact with each other through the package specifications, while

hiding the implementation details from other programs in the package body. The interfaces between the programs must be well defined and thought out, which help to ensure a higher quality product. With data abstraction, there is a single definition of a data object coupled with all the operations that can be performed on the object (Glass, 1992:187) This abstraction simplifies any changes made to the data object.

2.4.2 Structured Programming. Another maintainability technique used during development is structured programming which incorporates top-down design and the strict use of constructs. Programs should be modular; that is, they should be composed of small, hierarchical units and have a high degree of cohesion and low degree of coupling. Cohesion is the degree to which functions within the module are bound together, while coupling is the degree to which modules are dependent on each other. Structured programming has been shown to improve maintainability (Osborne, 1987:18).

2.4.3 Reviews and Audits. Reviews are conducted so management and the user can monitor the developer's progress. These reviews and audits assure a uniformity across the software, which is critical when someone other than the original programmer must maintain the code. The review determines if the design is meeting specific performance, design, and verification requirements and that the entire task can still be completed within the projected schedule and allocated budget (Glass, 1992:78). Audits are more closely related to inspections where SQA inspects code modules or test procedures for compliance with standards. During the early phases of the life cycle, audits can identify improper user defined requirements, design compromises, and poor documentation, problems which lead to poor maintainability. Reviews in the definition phase ensure that user requirements are clearly defined and understood by the development personnel. During the construction phase, quality assurance personnel review all documentation to determine that the documentation is complete, adequate, and meets the system's goals. In the implementation phase, quality assurance monitors and reviews all test plans and test reports, and finally reviews all documentation once again to ensure that the system has actually met its original goals (Wu, 1987:190-192). To ensure the system is supportable, and to become familiar with system, maintenance personnel should carefully read all documentation and participate in each review. The documentation and accompanying reviews are intended for technical people, not managers. Although SQA should evaluate the documents for standardization, the documents should be passed on the appropri-

ate personnel, (i.e. programmers and testers), who are able to evaluate the documents against the requirements. Although managers should attend the reviews to establish good working relationships with their corresponding development managers, it is the technical personnel who can make the greatest contribution to the quality of the system at a review.

2.4.4 Standards. Standards are essential for establishing a common maintenance environment. This common environment provides a common ground for reviewing another programmer's work, understanding another's code, and ultimately changing another person's program. This section is divided into development standards, coding practices, comments, and maintainability checklists.

2.4.4.1 Development Standards. In the Air Force, the development standard for weapon system software is DOD-STD-2167A, Military Standard for Defense System Software Development. The standard ensures that all contractors or organizations use the same uniform requirements in developing software, along with the means for establishing, evaluating, and maintaining quality in the software and its associated documents (DOD-STD-2167A, 1988: iii/iv).

2.4.4.2 Coding Practices. Good coding practices improve maintainability by providing a structure and framework in which systems can be developed and maintained in a clearer manner (Osborne, 1987:18). Management or software quality assurance places the requirements on programmers. An example list of practices may include:

- Language requirements - Use Ada or some other high order language.
- Code complexity - Use structured programming while avoiding undesirable language forms (programmer tricks which someone else might not understand).
- Modularity - Break a large procedure into smaller ones.
- Maintainability - Make the code easy to understand and change.
- Naming conventions - Use names that reflect the content or the variable or the structure of the program or data or both.
- Comments - Each subprogram must contain standard comments.
- Traceability - Each section of the code must trace back to the specifications (Glass,1992:90).

MIL-STD-1815A, Military Standard for the Ada Programming Language, specifies the form and meaning of Ada programs to facilitate supportability, portability and reusability. The standard provides specifications for:

- Declarations and types
- Names and expressions
- Statements
- Subprograms
- Packages
- Tasks
- Program structure
- Exceptions
- Generic Units
- Input and output (MIL-STD-1815A, 1983:I-IV)

When used effectively, coding practices provide a common ground for both the developers and maintainers to work.

2.4.4.3 Maintainability Checklists. David Sunday of the Northrop Corporation advocates a software maintainability checklist. The checklist provides an actual tool to evaluate the code from a maintainability perspective. Software designed against this checklist shows the following characteristics:

1. Consistency - correlate and contain uniform notation, terminology, and symbology while meeting the design objectives and requirements.
2. Testability - establish verification criteria and support evaluation of its performance, preferably at the module level.
3. Modularity - partition the software into reasonably sized parts, components, and modules; blocks of code are completely independent of all other modules within the program.
4. Descriptiveness - include information regarding their objectives, assumptions, inputs, processing, outputs, revision status, etc. The software avoids ambiguity and is readily understood. This includes code commenting and header information.

5. Changeability - must be able to change the information, computational functions, data storage, or execution, time can be easily accomplished.. You can move a module or a part of a module to simplify code reconstruction (Sunday, 1989:50-51).

2.4.4.4 Comments. Good comments also improve the maintainability of the system by providing readability and valuable information on the module including its design. This is important when someone other than the developer must modify the code. Comments should include the following:

- Why the code was implemented this way.
- How this module affects other modules.
- Any potential problems.
- When the changes were made.
- Who made the changes.
- What the changes were (Osborne, 1987:18).

2.4.5 Code Walkthroughs and Inspections. Humphrey describes a software inspection as a review of a programmer's or tester's work to find problems and improve quality by allowing them to recognize and fix their own problems before the system becomes operational. An inspection or walkthrough by peers can catch problems which the workers could not find, and which may not be discovered in testing. The inspection also ensures that both parties technically agree on the work and it meets predefined criteria. It's incorrect to call the process between development and maintenance personnel an inspection, since the term *inspection* could put the developers on the defensive. An appropriate term is formal walkthrough or just walkthrough.

The walkthrough follows these principles:

1. The walkthrough is a formal process with a system of checklists and defined roles. The roles include moderator, producer, reviewers, and recorder.
2. Checklists and standards are developed and tailored for each participant.
3. The participants prepare for the walkthrough in advance.
4. The purpose of the walkthrough is to discover problems.
5. The walkthroughs are for technical people, not managers.
6. The results are documented (Humphrey, 1990:171-177).

Because the walkthroughs involve personnel from three different organizations, the moderator should be from the PMO, who ensures that both the developers and maintainers are treated fairly and that the meeting doesn't turn into a finger pointing or shouting match.

There are three areas where walkthroughs can improve supportability: Design walkthroughs, code walkthroughs, and test walkthroughs.

Scheduling formal walkthroughs of the developed code with corresponding maintenance programmers improves maintainability in three ways. First, the maintainers are given the code to study before the inspection, which provides excellent training by familiarizing the maintainers with the code. Second, the maintainers provide an independent opinion, because they weren't the original programmers. Finally, the maintainers have a very high stake in the outcome: the better the inspection, the easier the maintenance.

The same rationale holds true for the testers. The testers from the maintenance organization should review the test cases. In fact, the maintenance testers should develop some of the test cases to get a feel for the system. Then, at the formal walkthrough, all test cases are reviewed by both the developers and the maintainers to look for faults. In either case, the results of the walkthroughs should be documented, and the recommendations should be implemented after the walkthrough. A follow up walkthrough is advisable to see if the recommendations were in fact implemented.

When coding ends, the focus shifts to supportability techniques in testing.

2.4.6 Testing. Wilma Osborne contends that a system's maintainability can be increased by improving the test plans and procedures (Osborne, 1987:19). A rigorous review and inspection process improves these plans and procedures. Improved test plans account for all requirements and conditions. Improved procedures increase the numbers of errors discovered and corrected, reduce the likelihood of future errors, and ensure a higher quality system. The testers in the maintenance organization can review or inspect the test plans, or even participate in the development of the test cases. The testers can also attend the Preliminary Design Review, the Critical Design Review, and the Test Readiness Review to evaluate the plans and procedures. By participating in the actual testing, the maintainers gain training on the system, a better understanding of how the system works, and how it is tested. The test procedures are passed on to the support organization for reuse or future regression testing (Guidelines, 1992:3-5).

During the error correction phase, the system's configuration management change process can be evaluated to ensure changes are properly identified, validated, and implemented. The software correction turnaround time should also be tracked so errors are identified and corrected in a reasonable amount of time.

Robert Arnold asserts that, in addition to standard software tests, a maintainability test should be performed throughout the coding cycle. A maintainability test requires that selected support programmers be able to make specified software changes within a specified amount of time (Arnold, 1987:27).

Once program management knows how to ensure supportability they need to know when to do it. The next section describes which techniques can be used in each phase of the development process.

2.5 Supportability Techniques in the Development Cycle

With software supportability, timing is just as important as method. The development cycle is broken into six phases: system requirements and design, software requirements analysis, preliminary design, detailed design, CSCI test, and system integration and test. The table on then next page shows what the maintainers can do in each phase of the development cycle.

Table 1
Supportability Techniques in the Development Cycle

Phase	Techniques
System Requirements and Design	Read System/Segment Specification, attend System Requirements Review
Software Requirements Analysis, Preliminary Design	Review Software Design Document (preliminary) and Software Test Plan, attend Preliminary Design Review.
Detailed Design	Review Software Design Document (detailed) and Software Test Procedures, attend Critical Design Review.
Code, CSU Test, CSC Integration Test	Code inspection by programmers. Joint SQA audits to enforce standards. Test case inspections. Evaluate CM during error correction. Maintainability tests.
CSCI Test, System Integration Test	Participate in testing. Evaluate CM during error correction. Maintainability tests. Physical Configuration Audit, Review Software Product Specification, Version Description Document, and manuals for completeness and correctness.

(Osborne, 1987:18; Sunday, 1989:50-51; Humphrey, 1990:171-177; DOD-2167A, 1988:12-13)

2.6 The IWSM Philosophy

Today, due in part to the tremendous world political changes and more austere military budgets, the process of weapon system acquisition and sustainment is evolving. As a result, military software system management practices are changing. This new management practice is Integrated Weapon System Management (IWSM).

IWSM is the management approach adopted by AFMC, a new command born from the merger of Air Force Logistics Command (AFLC) with Air Force Systems Command (AFSC). IWSM melds the strengths of both commands to create an integrated management philosophy to acquire and support weapon systems for its customer. IWSM is defined as: "the AFMC management philosophy for acquiring, evolving, and sustaining our products. It empowers a single manager with authority over the widest range

of decisions and resources to satisfy customer requirements throughout the life cycle of the product" (AFMCP 800-60,1993:2).

IWSM is recognized as a management philosophy, not a process. However, even as a philosophy, it necessitates changes in the responsibility levels, structures, partnerships, and processes of current weapon system acquisition and support organizations. This management approach creates a significant culture change within the military community. Now, a weapon system will be managed by a single individual who ties together systems acquisition and sustainment, two formerly separate functions. Conceptually, it is management of a single product for each element of a system (e.g., landing gear, engines, avionics) for that system's entire life cycle.

AFMCP 800-60 defines eight elements of IWSM which describe the important aspects of the philosophy. These aspects are closely interrelated and must be taken as a whole.

1. Quality Air Force. Creating an environment that inspires trust, teamwork, continuous improvement, and customer focus.

2. Cradle-to-Grave Management. Responsibility for all product decisions is given to a single manager. This manager carries this responsibility for the life of the system.

3. Single Face to the User. The single manager has complete authority over all program decisions and resources over the system's life cycle. Therefore, the user has a single individual to whom any and all concerns should be addressed.

4. Seamless Processes. Critical processes are integrated across the product life cycle. Therefore, no process seams should exist between organizations, locations, or program phases.

5. Empowered People. Authority and responsibility are assigned to the lowest level possible for efficiency. Empowered people have the ownership and responsibility for their respective products.

6. Common Sense Approach. If it doesn't make sense, fix it.

7. Integrated Product Development (IPD). IPD uses multidisciplinary teams to manage and integrate critical processes.

8. Product Focus. Strive to produce the optimum product with the goal of customer satisfaction. In the course of developing several trial programs, IWSM has created seven goals of weapon system management:

    1. Increasing the System Program Director's authority

    2. Creating a single business decision authority

    3. Inserting technology

4. Creating Integrated Product Teams (IPTs)
5. Maintaining management continuity
6. Building new partnerships
7. Consolidating Air Force acquisition (AFMCP 800-60, 1993:3)

These goals are incorporated into all managerial and technical processes. The System Program Director (SPD) is the single manager for the weapon system throughout its life cycle. The SPD manages two groups at different ends of the life cycle: the Development System Manager (DSM) and the System Support Manager (SSM). Also, the DSMs and SSMs manage several elements of their system(s) through Integrated Product Teams (IPTs). IPTs are multidisciplinary teams empowered with the responsibility for developing and/or supporting their respective system elements. Developers and maintainers are now brought together at program start to jointly manage system software until program cancellation or decommission.

Current PDSS planning and implementation processes are impacted by this new approach. The IWSM philosophy doesn't regulate or standardize a program's PDSS methodology. Therefore, individual weapon system programs decide, within the IWSM concept framework and applicable regulations, how best to plan for and implement PDSS for their weapon systems.

The IWSM philosophy is a way of looking at management relationships that encourages change to improve the operation of weapon system programs. This improvement involves consolidating expanded responsibilities into a single manager; revisiting and strengthening the relationships between many organizations; creating multidisciplinary process teams responsible for the products they design, build, and sustain; and maintaining management continuity throughout the weapon system's life cycle. The management of embedded software systems and, in particular, PDSS remains a vital element of the process.

2.7 Summary

This chapter presented the current literature available on ensuring software supportability and Integrated Weapon System Management (IWSM). The review presented ideas on planning for maintainability, acquisition requirements for supportability which can be levied on the contractor, and techniques for ensuring maintainability during development. The IWSM review presented the background which brought about the creation of the IWSM concept and how the concept is constructed.

With a background established in both IWSM and supportability, we now need to know what the actual IWSM software systems are doing to ensure supportability. The next chapter discusses the methodology employed to ascertain current and past supportability efforts in IWSM software systems, plus answer the investigative questions posed in chapter 1.

III. Methodology

3.1. Introduction

This chapter addresses the methodology employed to answer the investigative questions from chapter 1. A defined process was created to formalize the research. The first section covers the overall research process for this effort. The background of the population under study is then described. Next, the data collection method used in conjunction with the research process is outlined. Finally, the data analysis process is detailed.

3.2. Research Process

The research process involves building subject knowledge and data collection and analysis of observations. In order to develop a knowledge base and expertise on the subject matter, an extensive literature search is necessary. However, reading the pertinent literature will not totally answer or clarify questions on Air Force implementation of the principles. Interviews and surveys of the target population aid in focusing the research to answer the investigative questions.

The literature search provides the fundamental knowledge of software maintenance/support, software supportability, and the IWSM philosophy and its implementation. The literature will provide the guiding principles of software support and point out methods to design and plan for future software supportability. Of specific interest is any discussion of transitioning a developed software package into a support environment. Also, literature describing the tenets and principles of IWSM, as practiced by Air Force Materiel Command (AFMC), is necessary to conceptualize how Air Force weapon systems will be managed. Specifically, a concept is needed of how software support will fit into an IWSM management process.

Once a foundation is built, the research can proceed to the field to learn of real world implementation details.

Initially, various individuals in the Air Force community will be interviewed in person or by telephone to gather background knowledge to conduct the rest of the research. Their insight into military software support and the IWSM philosophy will guide our approach to the remaining research. Other literature may be suggested for review. Other points of contact for interviews and questionnaires will be obtained and contacted.

The System Program Directors (SPDs) and System Support Managers (SSMs), or their representatives, in the population of Product Centers, and Air Logistics Centers (ALCs), will be contacted initially to find the knowledgeable individuals to speak with. These individuals will be contacted to establish rapport and request support for more detailed information. Where face-to-face contact is impractical, a telephone interview will be arranged. These initial interviews will shape the follow-on questionnaire process. Each contact will be asked questions regarding their knowledge on efforts to improve software supportability during system development as it pertains to their weapon system program. Also, their general knowledge about the IWSM philosophy and its effect on their program's management structure will be surveyed.

Using the information from these initial interviews along with knowledge gained from the literature search, we will construct a questionnaire. We will review and revise the questionnaire to ensure adequacy and completeness. Individuals from the background interviews will be solicited to evaluate and comment on the questionnaire form's content. Once any revisions are completed, the questionnaire will be delivered to those knowledgeable individuals identified in the initial round of contacts.

3.2.1. The Questionnaire. The questionnaire (see Appendix A) will be the primary source of detailed data. It requests information from the individual identified in the program who manages the software development and/or support effort.

The questionnaire consists of five parts. The first part asks questions about the program's IWSM management structure. The second part asks questions about the program's planning efforts for PDSS. The third part asks how those PDSS plans are or were implemented during software development. The fourth

part asks questions about the program's transition from software development to support. The fifth part asks questions about operational software support.

3.3. Population Under Observation

Our population was selected to keep the research effort within a manageable scope. The target population is the 21 Air Force weapon systems selected as pilot programs to implement the IWSM philosophy. In addition, several Command, Control, Communications, Computer, and Intelligence (C4I) programs are included as IWSM pilot programs. These programs can be characterized as mission support systems which are software intensive. The population is split into two categories: those weapon systems which are operational and those still under development. This split may point out differences in the system management approaches between the categories. The weapon systems under development may have management structures which are more flexible to change. It is our expectation that with limited resources and schedule, this thesis may address only a few of the weapon system programs. Following is a brief description of the programs who were sent questionnaires.

3.3.1. IWSM Weapon System Programs

3.3.1.1. Automatic Test Systems (ATS). Ongoing effort to support existing weapon systems with automated test equipment and survey DoD logistic support centers for commonality between ATS hardware and software test sets.

3.3.1.2. B-2A. Development, production, and supportable deployment of a four-engine, low-observable, flying-wing type of strategic penetrating bomber, designed specifically to elude enemy air defenses. [Mehuron, Jan 1993:56]

3.3.1.3. C-130. Support for operational military cargo aircraft including software support of operational flight program.

3.3.1.4. Airborne Warning and Control System (AWACS) (E-3).

Development of a major upgrade for the AWACS surveillance and battle management aircraft, including radar system upgrades. [Mehuron, Jul 1992:34]

3.3.1.5. F-15. Development and support of various programs including F-15E dual-role fighter, subsystem, and support equipment programs. [Mehuron, Jan 1993:57]

3.3.1.6. F-16. Ongoing development and support for a single-engine, lightweight, high-performance, tactical fighter with an air-to-air and air-to-surface multirole capability that can be deployed from the continental US to any trouble spot in the world. [Mehuron, Jan 1993:57] Currently involved in foreign military sales (FMS).

3.3.1.7. F-22. Development of next-generation Air Force air superiority fighter including advanced propulsion, flight controls, fire controls, significant avionics integration, designed supportability characteristics, et al. [Mehuron, Jan 1993:57]

3.3.1.8. F-117A. Development and production of major modifications to F-117 avionics including an upgrade to forward-looking infrared (FLIR) and improvements to the navigation system. [Mehuron, Jan 1993:56]

3.3.1.9. Navstar Global Positioning System (GPS). Deployment and support for a system that provides 24 hour, all-weather, worldwide, space-based radio navigation capabilities for military and civilian users with extremely accurate three-dimensional position information. [Air Force Magazine, Aug 1992:34]

3.3.1.10. ICBM. Modernization programs for the Minuteman intercontinental ballistic missile.

3.3.1.11. Joint Surveillance and Target Attack Radar System (Joint STARS). Development of a joint USAF-Army sensor which integrates a side-looking multimode radar into an E-8A platform to create a targeting system able to detect stationary and moving ground-based objects. [Air Force Magazine, Jul 1992:35]

3.3.1.12. LANTIRN. Production of a two-pod navigation and targeting system for night, under-the-weather ground attack by F-15E and F-16C/D aircraft. Provides video display of flight

path terrain, terrain-following radar (TFR), infrared target detection and tracking and laser designation and range-finding. It is used for precision munitions deliveries. [Mehuron, Jan 1993:57]

3.3.2. C4I Programs. The C4I programs were added to the IWSM pilot programs after the weapon systems. The C4I programs differ from the weapons systems in that C4I programs are primarily computer hardware and software as opposed to a complete weapon system such as an aircraft. The scale of a C4I system (cost, schedule, and effort) is also smaller than a weapon system. The three programs which were sent questionnaires are the Combat Ammunition System (CAS), the Wing Command and Control System (WCCS), and the Atmospheric Early Warning System (AEWS).

3.4. Data Collection

The data will be broken into two subsets, software intensive weapon systems under development and those in operation requiring software support. Initial phone contacts and questionnaires are used for two purposes. First, the knowledgeable person in the organization, that is, the SPD, the SSM or their designated representative is determined. This individual should be articulate about addressing software supportability concerns in development and software support performance in their current IWSM structure. Secondly, we must determine what, if any, software system management techniques are used. The individual will be asked to participate in completing a questionnaire.

3.4.1. Data Assimilation and Sampling Risks. Once the questionnaires are sent, a suitable period will be allowed for responses to return. If responses aren't returned by that time, follow-up phone calls will determine their status. This phase of the thesis effort presents the most risk. If the contacts choose not to or are unable to respond within a reasonable time, then a potential source of data may be lost. The analysis may suffer due to insufficient sample size and poor representation of the population. Also, other thesis tasks can run in parallel while awaiting responses.

3.4.1.1. Response Processing. Upon receipt of each response, the following steps take place:

1. Assign a control number to each response for tracking purposes. The control number dissociates the response with any particular individual or program.

2. Review each answer for quality. Although the questionnaire introduction states that answers shouldn't be restricted to "yes" or "no", we cannot prevent such an occurrence. Several questions require some elaboration.

3. Develop clarification questions from incomplete or ambiguous answers as required. Follow-up phone calls will resolve these questions.

4. Separate software supportability planning data from post-deployment software support data. Each subset provides material to develop its respective part of the management guidelines.

5. Cross reference IWSM management information with program maturity, i.e., development or operational. Again, this data will support development of the overall management guidelines.

3.5. Analysis and Observations

Analysis remains the most difficult part of any research effort for the novice researcher (Emory and Cooper, 1991:16). The objective of our analysis is to infer any characteristics or trends of IWSM organizations and their methods of managing software intensive systems. The management guidelines will summarize the analysis results.

The data gathered from the literature review, background interviews, and the questionnaire form the analysis domain. A brief description of each data element's utility follows.

3.5.1. Literature Review. The review furnishes not only the basis of software support knowledge but also the principles of managing a software support effort. These principles supply the initial guidelines.

3.5.2. Background Interviews. The interviews set the stage for understanding the principles of IWSM and USAF software support planning and management practices. This information aids in the development of the questionnaire.

3.5.3. *The Questionnaire.*

The questionnaire organizes data in three areas: the weapon system program's IWSM management structure, software supportability planning efforts, and post-deployment software support performance.

3.5.3.1. *IWSM Management Structure.*

This section establishes background on each program's organization. Of interest is their development of Integrated Product Teams (IPTs) and how they address planning and/or support. This section helps distinguish system management methodologies without inferring their success or failure. The derived organization characteristics provide an explanation of the software supportability planning approach.

3.5.3.2. *Software Supportability Planning.*

This section measures the level of planning performed during the weapon system's development phase. These measurements serve as a metric to judge level of planning versus the perceived software support performance.

3.5.3.3. *Post-Deployment Software Support.*

Here, ordinal data is gathered to describe the success or failure of the software support effort. Inferences may be drawn relating the management techniques and support planning efforts to software support program success. Note that some weapon system programs under study haven't yet fielded operational systems. Therefore, we can only predict the outcome of any future software support effort from their supportability planning and IWSM management structure.

3.6. *Management Guideline Development*

The ultimate objective of this thesis is to develop a set of management guidelines for enhanced future software support under the IWSM philosophy. The guidelines are intended for use by system acquisition and software system managers of software intensive systems. The guidelines address management principles to apply during system development and operational support to prepare for or improve software support.

The development process requires four steps:

1. Draft initial guidelines based on the literature review. This task was accomplished while we awaited questionnaire responses.

2. Update the guidelines with new information from questionnaire response analysis.

3. Submit final drafts to several experts knowledgeable in the areas of software support and representatives from each IWSM program which submitted a questionnaire response.

4. Gather comments and incorporate them into a final set of guidelines.

During this process, conflicts may arise between authors, questionnaire answers (e.g., management approaches), and guideline comments. To resolve the conflicts, we need to understand the basis of the conflicting elements. For example, conflicts in management approaches may be a result of differing assumptions. We'll attempt to point out those assumptions, validate them if possible, and suggest alternative guidelines based on the particular assumptions.

When this thesis and the guidelines are completed, the results will be submitted to SAF/AQ for consideration in their ongoing effort to establish management guidelines for software intensive systems.

3.7. Summary

This chapter described the methodology for answering the investigative questions posed in chapter 1. The methodology employs a literature review, an initial telephone interview (with possible follow-ups), a questionnaire, and the construction of management guidelines using the accumulated data. Our target population description indicates the scope of our study. The process of data collection and assimilation was described with its inherent risks. Finally, the analysis and observation methodology was related. Chapter 4 contains the results of our analysis.

IV. Results and Analysis

4.1 Introduction

Chapter 4 describes the results obtained from the gathered data. First, the data itself deserves some discussion in regards to its characteristics, quality, and applicability to comparison. Per the organization of the questionnaire, the data can be grouped to study the different stages of PDSS: planning, implementing supportability during development, transition from development to support, and PDSS implementation. The data collected and studied at each stage for aircraft, electronic, space, and C4I systems. Our observations and comparisons supplement the management guidelines with recommended techniques.

4.2 Data

4.2.1 Data Characteristics. Data was collected from several weapon system and C4I programs across Air Force Materiel Command (AFMC). The data represents responses from AFMC Product Centers and Air Logistics Centers (ALCs), the former involved with weapon system procurement and the latter with sustainment. Under the IWSM philosophy, Product Center and ALC personnel are managed under a single System Program Director (SPD) to procure and maintain weapon system software.

Although the questionnaire requested elaboration on "yes or no" type questions, we received a mix of terse and detailed responses. The elaboration was intended to qualify any yes or no answers.

4.2.2 Data Quality and Quantity. Not all IWSM pilot programs and C4I programs could be reached to deliver questionnaires. Of those programs which were given questionnaires, not all returned responses. Of fifteen questionnaires sent, eight were completed and returned. This is a return rate of approximately 54%. One of the original IWSM programs, Automatic Test Equipment (ATE), is not currently developing or supporting software. A questionnaire for that program was deemed inappropriate, and, instead, an in-depth interview was conducted. The IWSM focal point at Human Systems Center could not be reached, therefore, human system programs weren't considered in our analysis. Although

several attempts at contact were made, not all aircraft system programs could be reached or a software point of contact established. Of the seven programs which didn't respond, attempts were made to remind the volunteers of the needed information. Finally, a cutoff date was established in order to proceed with the analysis. If a response came late, it wasn't included in the analysis to prevent bias.

As shown in figure 2, the percentage of total responses returned by aircraft, electronic, and space systems were relatively even. Only one of the three C4I programs surveyed responded. One data point on C4I programs is not sufficient to characterize them. Therefore, any conclusions drawn on C4I PDSS planning under IWSM would be speculative at best.

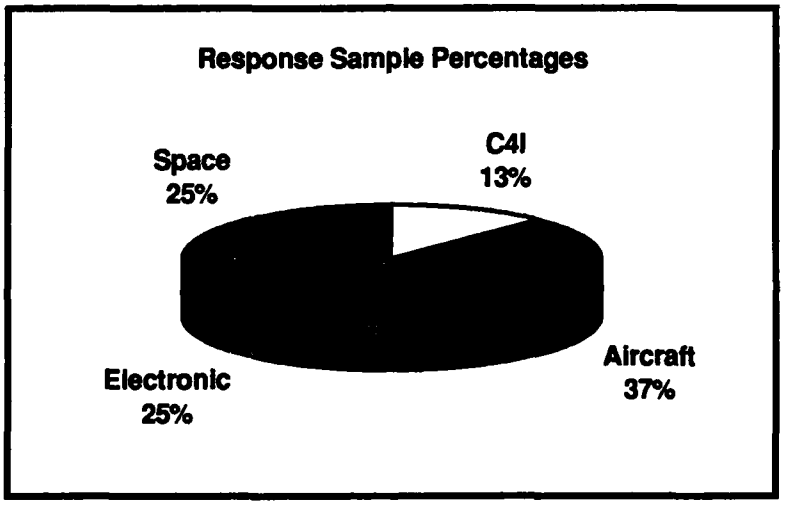


Figure 2 Breakdown of Respondents

4.2.3 Data Manipulation. The respondents' data was recorded using a personal computer spreadsheet application. Responses to each questionnaire question were represented in individual spreadsheet cells. However, only a binary entry (e.g. yes or no) could be entered into the spreadsheet. Qualitative answers could not be represented but they were used in addressing the investigative questions. Separate spreadsheet regions were defined to separate results from the different weapon system categories. From this point, data could be formatted to produce charts used in analysis.

4.3 Program Organization Under IWSM

General IWSM information was collected on each program. The following summarizes the various management principles used for software development and/or support efforts.

Figure 3 clearly shows that software is a factor in 72% of the total Integrated Product Team (IPT) efforts. Therefore, it is a significant factor in managing a weapon system. Those IPTs which don't directly involve software development/support are functional support areas such as contracting, site activation, and system testing. 88% of all programs surveyed integrate their functional elements with their IPTs. Of these programs, one organized its IPTs around boards and working groups versus system elements. Also,

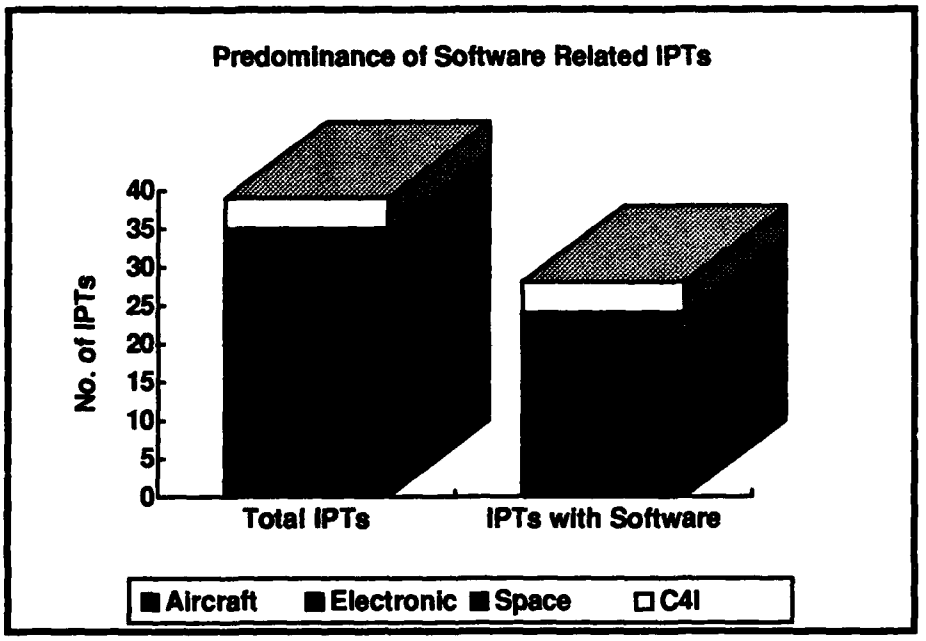


Figure 3 Total IPTs vs. Those with Software Efforts

one program had a single software IPT for all software efforts. The remaining program separated some functional areas into stand alone IPTs. Of these programs, one also had a single software IPT.

In 88% of the programs surveyed, personnel were assigned in a matrixed or matrixed/dedicated fashion to their IPTs as shown in figure 4. One of these programs dedicated their personnel to their IPTs.

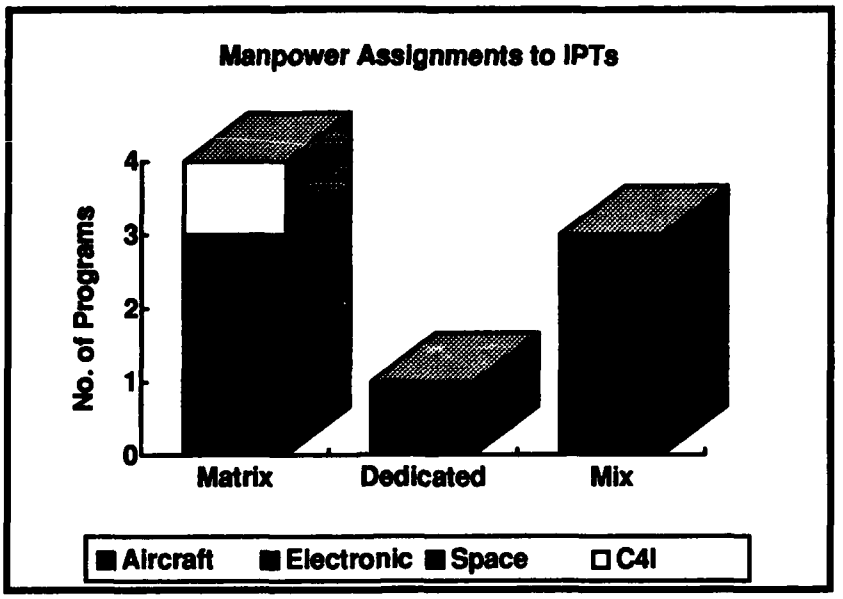


Figure 4 Management Organization of IPTs

With the majority of IPTs involved in managing software development or support, are those efforts managed separately or coordinated with each other? Managers can foresee resource, schedule, budget, and requirements conflicts of concurrent software efforts which are coordinated with one another. Although there may be reason to manage some efforts separately, doing so without an integrated perspective may complicate future software supportability. Figure 5 indicates 63% of the efforts are coordinated with each other. Of those programs with separate management efforts, one respondent felt their new team

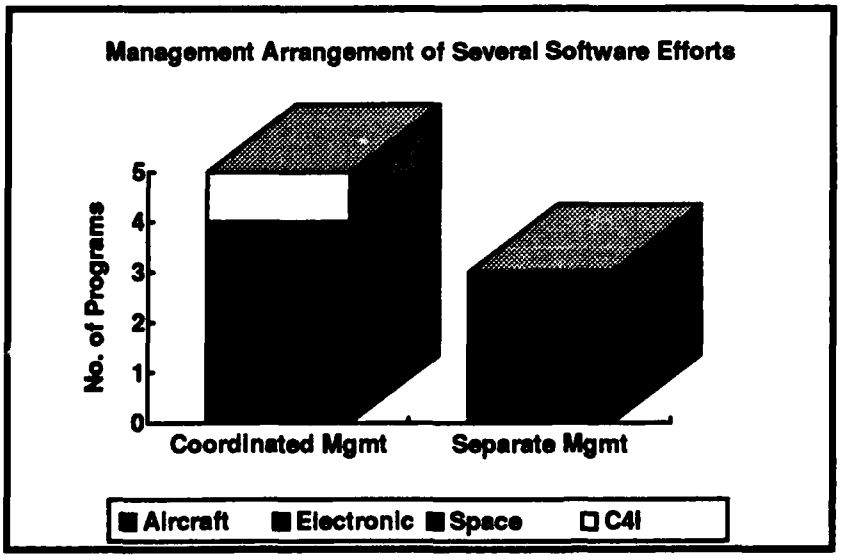


Figure 5 How the Programs Manage Their Software Efforts

With the majority of IPTs involved in managing software development or support, are those efforts managed separately or coordinated with each other? Managers can foresee resource, schedule, budget, and requirements conflicts of concurrent software efforts which are coordinated with one another. Although there may be reason to manage some efforts separately, doing so without an integrated perspective may complicate future software supportability. Figure 5 indicates 63% of the efforts are coordinated with each other. Of those programs with separate management efforts, one respondent felt their new team

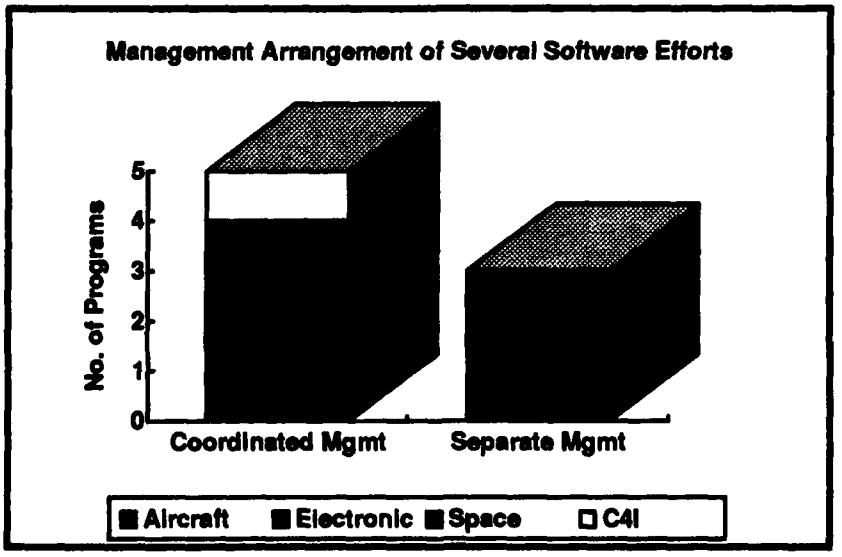


Figure 5 How the Programs Manage Their Software Efforts

4.4 Planning for PDSS

All but one of the responding programs have a Computer Resources Lifecycle Management Plan (CRLCMP). The CRLCMP addresses PDSS planning including the number and type of personnel required to perform PDSS, and the equipment and environment required. The remaining program used a system maintenance plan (focusing on more than software support) such as an Integrated Logistics Support Plan (ILSP). Three of the programs surveyed had both a system maintenance plan and CRLCMP which addressed software support. In this case, the CRLCMP was the central software planning document. The companion system maintenance plan was sparse in details in 50% of these programs, thus deferring to the CRLCMP.

Other valuable resources in PDSS planning are configuration management and software quality assurance (or quality assurance in general) personnel. Figure 6 shows 88% of the programs surveyed employed configuration management and 38% employed software quality assurance in their planning efforts. 12% of the programs report using neither function in PDSS planning.

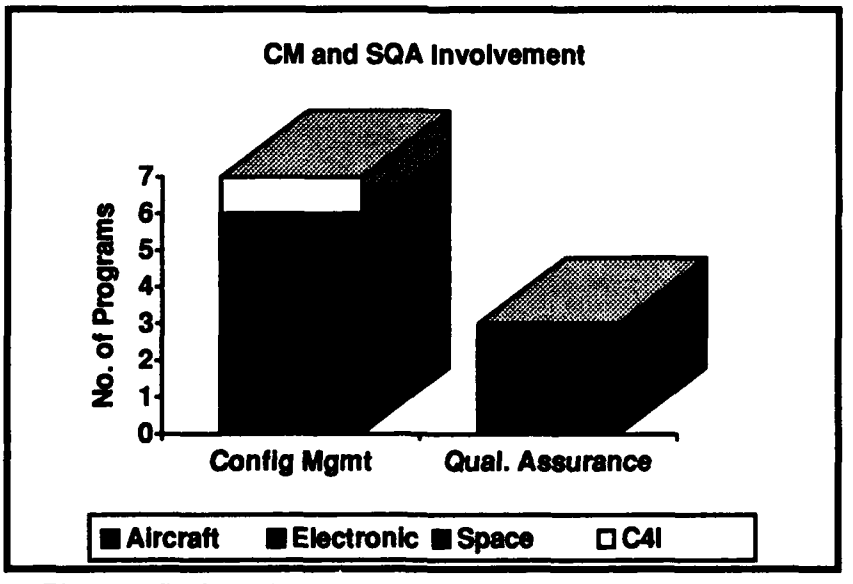


Figure 6 Configuration Management and Software Quality Assurance Involvement in Implementing PDSS Plans

4.5 Implementing PDSS Planning During Software Development

The primary materials software support personnel have to work with are documentation and software code. As shown in figure 7, the SSM organization participated in software development reviews and audits in 75% of the programs surveyed. Software support personnel were *able* to inspect software code in 63% of the programs. 25% of the programs reported that although software support personnel are able to inspect code either they don't do it or "rarely have the time". Another 25% of the programs had peculiar limitations in the ability to review software code as it was being developed. One program reported the software components had to be assembled before inspections could take place. The other program could inspect only the application type and size (e.g., number of software lines of code) of the code.

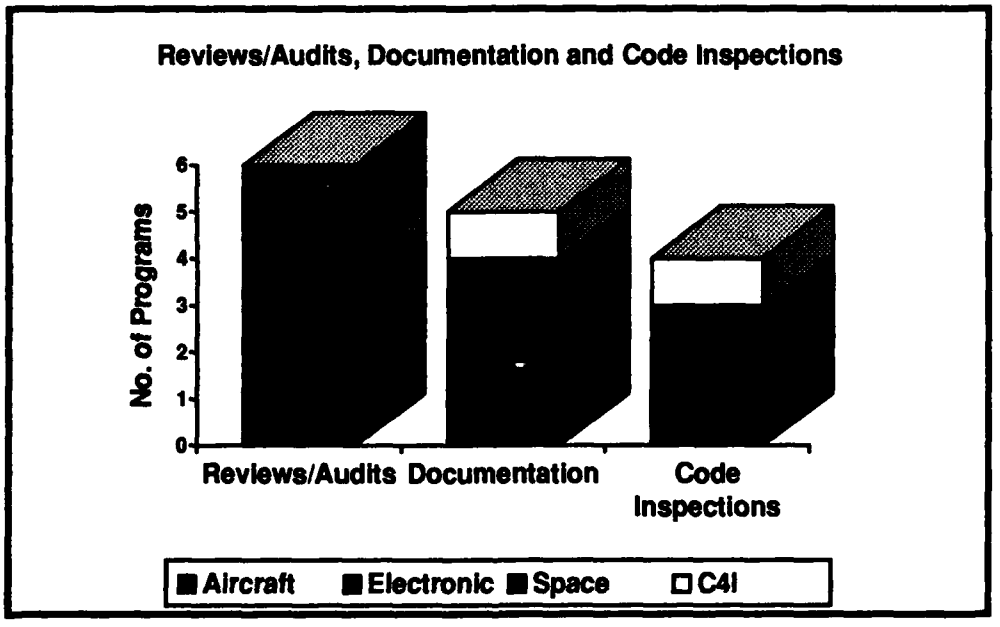


Figure 7 Participation in Software Reviews, Audits, Documentation Reviews and Code Inspections

Only 25% of the programs performed software supportability testing as shown in figure 8. Software support personnel participated in 75% of the programs' overall software testing efforts. This participation provides the supporter the opportunity to see how the software is tested. The participation level contributes to the learning experience. Software supporters participated by witnessing and hands-on support in 75% and 50% of the programs, respectively.

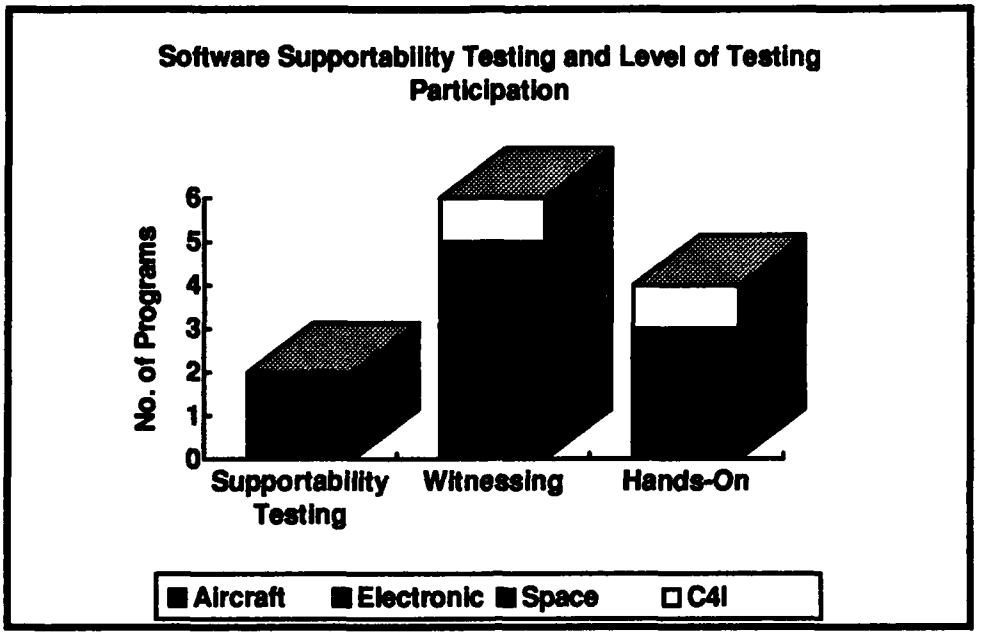


Figure 8 Level of Supportability Testing Performed and Level of Software Support Personnel Participation in Testing

4.6 Transitioning from Software Development to Software Support

The information in this section applies to operational systems only. Only 50% of the programs responding to the survey have transitioned from a software development to support environment.

All four operational programs report that their respective transition plans were followed. 60% of the programs[1] received training from the software developer. As shown in figure 9, three programs reported a routine transition while the other two reported a difficult transition. The three "routine" programs received training, however, the two "difficult" programs were split on the training provision. These two programs have mixed organic/contracted PDSS organizations. One program had difficulty transitioning to organic support. Necessary software support requirements, including facility, location, security, and personnel training and qualifications, were difficult to define exactly. The other "difficult" program is

[1] One of the four operational programs was divided into two distinct programs. These programs represent different models of the system whose support plans differ.

managing several dozen concurrent software development and support efforts. It is interesting to note that the three "routine" programs have total Contractor Logistics Support (CLS) arrangements.

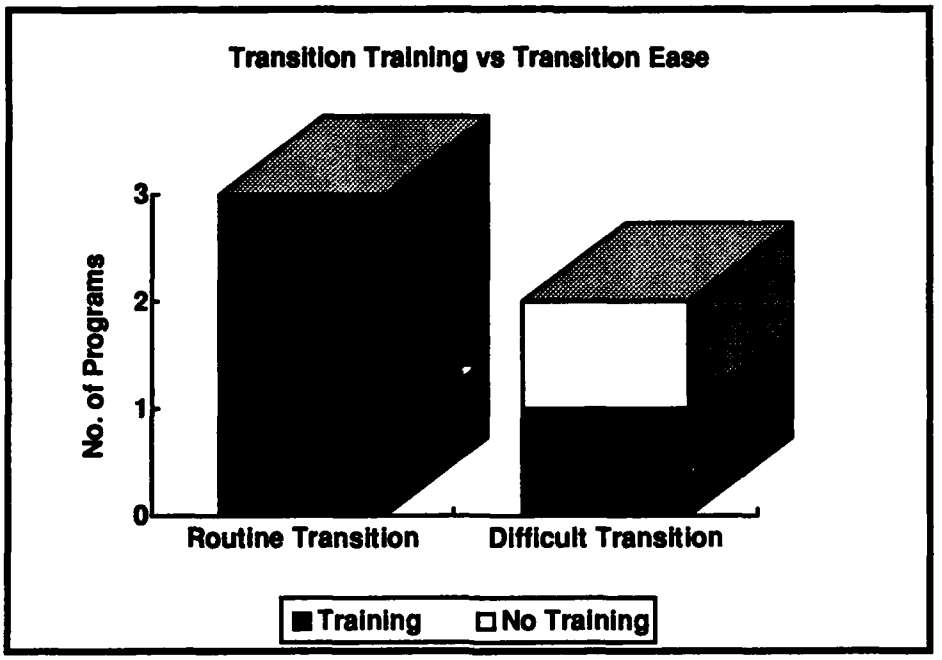


Figure 9 Impact of Training on Ease of Transition

4.7 PDSS Stage

The results during the PDSS phase of the operational programs, as seen in figure 10, are much the same as during transition. Three of the programs have a CLS organization, while the remaining two have a mix of contract and organic support, and no programs have solely organic support. The computer resources chief of one of the "difficult" programs, in a move to minimize the difficulty between development and support efforts, authored a program operating instruction to address computer resources support as a program institution. Where many organizations are involved, as discussed in paragraph 4.3, such an operating instruction serves as a common baseline procedures document.

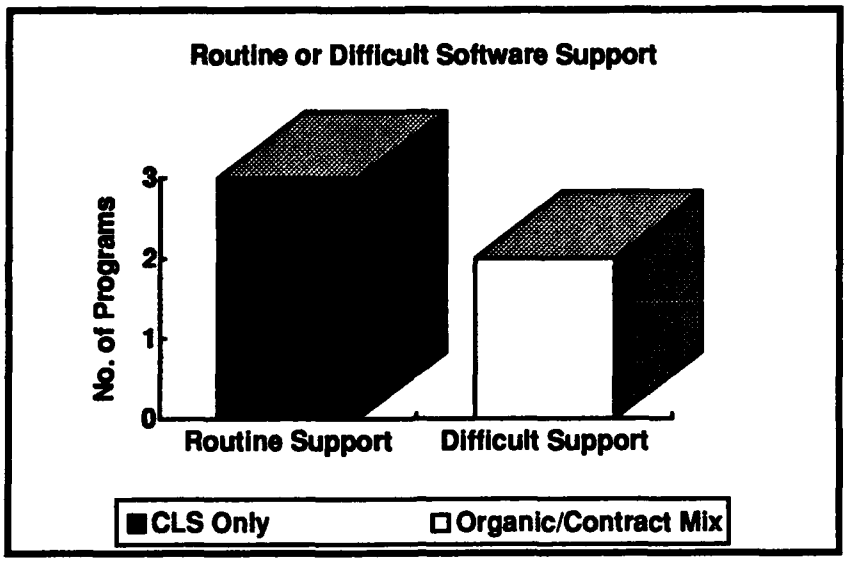


Figure 10 Routine or Difficult Support Effort

4.8 Guidelines for Ensuring Software Supportability in Systems Developed Under IWSM

The goal of this thesis was the development of management guidelines for weapon system managers in an IWSM environment who must field and support software intensive systems. The formation of the guidelines followed this process:

A. The initial draft of the guidelines was based on the literature review from chapter two. This task was accomplished while awaiting questionnaire responses.

B. Any additional information from questionnaire response analysis was incorporated into the guidelines. The changes to the guidelines consisted of software maintainer involvement in the development of the Software Configuration Management Plan and the Software Quality Assurance Plan.

C. The second drafts were submitted to several experts knowledgeable in the areas of software support. Four experts in the field of software supportability were invited to review the draft guidelines. Only three accepted the invitation. Of these three, only two returned recommendations. One expert is assigned at SAF/AQK, and is responsible for producing the *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems and Management Information Sys-*

tems. The other expert is an instructor at the Air Force Institute of Technology, with an extensive background in software support, and the instructor for the Institute's Software Maintenance and Generation course. The recommendations from the experts include: elaborate why the techniques presented in the guidelines are unique to the IWSM programs, explain why a separate PDSS plan is needed besides the CRLCMP, include a discussion on improving supportability through the use of commercial off-the-shelf and reusable software, and include supportability requirements in the software developer's Memorandum of Agreement.

D. The final set of guidelines is a combination of the expert opinion from the literature review, additional information provided by the survey responses, and recommendations from the experts who validated the guidelines.

4.9 Findings

4.9.1 Investigative Question #1. What maintainability planning have the software support management personnel done in current software system development? 80% of the program management personnel with systems currently under development indicated that software support planning was documented both in the Computer Resources Lifecycle Management Plan (CRLCMP) and a separate system maintenance plan. These programs used the separate plan only to document the personnel requirements, while the CRLCMP was used to document the equipment and facilities requirements. Interestingly enough, the other 20% who used either a system maintenance plan *or* the CRLCMP provided a more complete plan in terms of personnel, equipment, supportability techniques, and transition planning than the systems which had *both* a maintenance plan and a CRLCMP. No reasons were given.

4.9.2 Investigative Question #2. Are the software maintainers participating in the development of the software to improve maintainability? 80% of the survey participants indicated that software maintenance personnel worked with configuration management personnel to ensure supportability, but only 40% worked with software quality assurance to make supportability a goal. 80% of those surveyed said the maintainers reviewed (or were given the opportunity to review) all documentation devel-

oped before the formal review, and also attended (or were given the opportunity to attend) the actual review. 40% of the systems surveyed said the maintainers inspected the code as it was developed. All but one of the maintainers participated in software testing, either hands-on or witnessing the actual tests.

4.9.3 Investigative Question #3. For systems already in the support phase:

Was maintainability planning done by the SSM personnel during development? 67% of those surveyed indicated that the only support planning was documented in the CRLCMP. All systems who used the CRLCMP documented the personnel and equipment requirements, while only 50% documented the supportability techniques and transition plan. The one system that used a separate maintenance plan documented only the supportability techniques.

A. Did the software maintainers participate in the development of the software to improve maintainability? All of the participants said the support personnel worked with configuration management personnel to ensure supportability, while only 50% worked with software quality assurance. All of the maintainers attended at least some of the reviews and read the documents prior to each review. None of the maintainers inspected the code as it was developed, although 75% of the maintainers were given the opportunity. 67% of the support personnel participated in software testing.

B. How satisfied are the software maintainers with the system they must maintain? 66% of the survey participants are satisfied with the follow-on software support. One of the primary reasons for this satisfaction is that the development contractor is also the follow-on support contractor, which simplified any transition, expertise, or training problems. On the down side, one survey participant described unsatisfactory support was as "uncontrollable", primarily because of the many different configurations (34) present in this weapon system.

C. What do the software maintainers wish could have been done differently during software development to make their software maintenance easier? The respondents had several recommendations. Some desired changes to their programs' software development effort include:

- Try to retain common software configurations between systems,
- Require the software developer to use modular code,
- Use standard software libraries to minimize redevelopment of common algorithms,

- Institute common coding practices between the development and maintenance organizations,
- Use a common software development process across all organizations,
- Ensure the required software documentation is deliverable under a development contract, including documentation produced by a subcontractor, and,
- Educate the contractor as to why all of the above is needed by the government.

D. Is there any correlation between early and effective maintainability planning and implementation and the satisfaction of the software maintainer? 38% of the support planning was placed in the CRLCMP instead of (or along with) a separate system maintenance plan. The CRLCMP primarily covered the personnel and equipment requirements, but omitted the supportability techniques and transition planning. This shallow software maintenance planning combined with relatively few supportability techniques implemented during development resulted in software which was difficult to support. The system with the most complaints about support did no planning for supportability techniques or transition. In this system, the maintainers performed no software inspections and participated in none of the testing during development. In contrast, the system with the least complaints planned for personnel, equipment, supportability techniques, and transition. The maintainers for this system worked with configuration management and software quality assurance to ensure supportability. The maintainers also attended reviews, inspected the software, and participated in testing.

4.9.4 Investigative Question #4. How do the current plans compare with past plans? The current plans have a separate software maintenance plan, whereas the past plans use only the CRLCMP. The systems under development also used walkthroughs more than the operational systems, but less than 40% of the development systems used walkthroughs, which was unexpected.

4.9.5 Investigative Question #5. What planning techniques do the experts recommend for improving maintainability? The techniques are located in sections 2.2 through 2.5 of the literature review. The techniques stress up-front planning for personnel and equipment requirements, and the responsibilities of a support organization. The experts recommend documenting all training, documentation, and quality assurance requirements for the development contractor. The techniques for implementing supportability planning during development include: reviews and audits, walkthroughs, and test par-

ticipation. These recommendations became the basis for the guidelines for ensuring software supportability.

4.9.6 Investigative Question #6. How do these techniques compare with techniques used in past and current weapon system software development? The questionnaire mirrored the literature review, so, in effect, we were surveying the various system offices for comparison with the experts' opinions of supportability techniques. While all of the systems surveyed used at least some of the expert recommendations, only one system used all of the recommendations for planning and supportability techniques.

4.9.7 Investigative Question #7. Can past maintainability plans be combined with current efforts and expert opinion to provide a set of guidelines for ensuring future maintainability efforts? The initial set of guidelines from the literature search were combined with the questionnaire responses and recommendations from the evaluators to produce the final versions of the guidelines. The guidelines are included in Appendix B.

4.10 Summary

This chapter provided analysis of the data from the survey responses, followed by the steps taken to construct and validate the guidelines for ensuring supportability. Each of the investigative questions from chapter 1 was answered. The next chapter contains the research results and recommendations for follow-on research.

V. Conclusions

5.1 Introduction

The goal of this thesis was to determine how the Software Support Manager can plan for Post Deployment Software Support (PDSS) and how the software maintainers can participate during development to improve supportability. The result of the research was a set of guidelines for ensuring supportability in software systems developed under the Integrated Weapon System Management (IWSM) concept. First, a detailed literature search was performed to determine what the experts recommend to ensure software supportability. Then, selected IWSM software systems were surveyed, based on the investigative questions from chapter one, to determine what was done in past and present systems to ensure supportability. The results of these first two procedures were combined to form an initial set of guidelines for ensuring supportability. This initial set of guidelines were validated by a panel of software supportability experts. The research results, as well as recommendations for follow on-research are presented in the next two sections.

5.2 Research Results

The research revolved around three phases, which, when completed, provided us with the guidelines for ensuring software supportability:

A. The in-depth literature search provided a detailed background on planning for software maintainability and presented techniques which could be used during development to ensure supportability in the software. The literature review stressed up-front planning for personnel, equipment, supportability techniques, and transition from development to support. The review also included these supportability techniques which could be used during development: reviews and audits, inspections, and test participation.

B. A survey on planning for PDSS and ensuring supportability within an IWSM organization was developed which would answer the seven investigative questions from chapter one. The survey of past and present software supportability efforts allowed us to compare actual Air Force practice with the expert opinions from the literature review. We found that the operational systems only used the Computer Resources Lifecycle Management Plan (CRLCMP) and documented only personnel and equipment requirements. The developmental systems used a separate maintenance plan along with (or instead of) the CRLCMP. Although all the systems used at least some of the expert recommendations, only one system used all the recommendations. The systems under development also used software inspections more than the operational systems.

C. The initial guidelines for supportability were presented to a panel of software support experts for validation. The recommendations included: clarifying why the techniques benefit IWSM programs more than non-IWSM programs, discussing commercial off-the-shelf software and reusable software, and preparing Memorandums of Agreement for supportability with the software developer.

The research culminated with the final version of the guidelines for ensuring supportability. The first research objective is answered in chapter 2 of the guidelines, *Planning for PDSS*. The chapter stresses up front planning for personnel, equipment, contractor requirement, and transition planning. The second research objective is covered in chapter 2 of the guidelines, *Ensuring Supportability During Development*. This chapter recommends attending reviews and audits, holding software inspections, and participating in tests.

5.3 Recommendations for Further Research

5.3.1 Supportability in Non IWSM Programs.

This thesis focused on programs using IWSM. The questionnaire in this thesis can be used to survey the non-IWSM programs. The non-IWSM programs have separate development and support managers. Non-IWSM and IWSM programs can be compared to determine:

1. Which management style did more up front planning for PDSS?

2. Which management style had more participation by the software maintainers during development to ensure supportability?

3. How satisfied are the maintainers with the developed software they received under each management style?

5.3.2 Supportability Metrics. The follow-on research would survey operation systems to determine which factors correlate with supportability and determine metrics to measure these factors. Then the research should determine which values correlate with easy and hard support. These metrics could then be applied to systems under development to determine the future supportability of a software system.

5.3.3 C4I Programs. The single program which implemented all of the expert recommendations was a Command, Control, Communications, Computer, and Intelligence (C4I) system. However, it was the only response that we received from the C4I community. The questionnaire and data accumulated in this thesis can be used to survey the rest of the C4I programs to determine if there is any trend among the C4I programs. If any type of trend exists, can the findings be applied to the weapon systems programs?

5.3.4 Validate Research Results. Of all IWSM programs (20 in all) and C4I programs, only eight responded to the questionnaire. A larger sample, which could include additional programs, should be obtained to validate the results of this effort.

5.4 Summary

With the excessive cost of software support coupled with shrinking budgets in today's environment, the Air Force needs to focus on improving the supportability of the software it develops or acquires. The Air Force has already taken a large step in improving the supportability of the software systems by implementing the IWSM concept. The single manager will want to develop software that is easy to support. The guidelines presented in this thesis will enable the Software Support Manager to plan for PDSS early in the lifecycle, implement techniques for ensuring supportability during development, and reap the benefits during the system's operational lifetime.

Appendix A
The Questionnaire

QUESTIONNAIRE TO SUPPORT DEVELOPMENT OF SOFTWARE SUPPORT GUIDELINES IN SOFTWARE SYSTEM DESIGN AND SUPPORT UNDER INTEGRATED WEAPON SYSTEM MANAGEMENT

INTRODUCTION:

We appreciate your participation in our effort to develop a set of software support guidelines for software system managers under IWSM. Your inputs are an important part of our thesis research effort and the key to its success. We are working with SAF/AQKS to study the impacts IWSM has on preparing for and managing post deployment software support (PDSS). One thing we want to make absolutely clear is that we are only interested in obtaining a "big picture" perspective of the software support planning and management processes based on the initial 21 IWSM pilot programs. Each individual input we receive will be considered confidential, and there will be no ties to specific programs or individuals. We will gladly share with you the results of our thesis project if you would like us to do so.

The survey should be answered by someone in the program office who manages the software support effort, not technical personnel. The survey consists of five parts. The first part asks questions about your program's IWSM management structure. The second part asks questions about your program's planning efforts for PDSS. The third part asks questions about how those PDSS plans are implemented during software development. The fourth part asks questions about the transition from software development to support. The fifth part asks questions about operational software support. The entire survey should take no more than 45 minutes to complete.

Please remember that you are not restricted to a simple "yes" or "no" answer. Any elaboration will be greatly appreciated. If an answer to a question is unknown, ask if there is an individual in your organization who may be able to answer it.

If your system is not yet operational, please check here _____ and complete only parts I through III.

Part I: IWSM Management Structure

1. Please list each of your program's Integrated Product Teams (IPTs) and briefly describe the purpose of each.

2. Which functional areas support these IPTs? (e.g., configuration management, engineering, finance)

3. Which IPT(s) involve(s) the development and/or support of software?

4. For each IPT listed in #3, are software development/support personnel (i.e., government civilian, military, support contractor) matrixed, dedicated, or a mix?

5. Do the software development/support personnel participate in your program's Computer Resources Working Group (CRWG)?

6. Are all the software efforts managed in a coordinated manner or separately?

7. Do the software development/support personnel within the System Development Manager's (SDM) and System Support Manager's (SSM) organizations meet regularly? If so, how frequently and by what means?

Part II: Planning for PDSS

1. Is there a maintenance plan for the system?

2. If one exists:
   A. Does the plan include the number and type of personnel required for software support?

   B. Does the plan identify the equipment and environment required to support the software?

   C. Does the plan address techniques for ensuring software supportability in the development phase?

   D. Does the plan address the transition from software development to software support?

3. If no system maintenance plan exists, does the Computer Resources Lifecycle Management Plan (CRLCMP) address PDSS?

4. If so:
   A. Does the plan include the number and type of personnel required for software support?

   B. Does the plan identify the equipment and environment required to support the software?

   C. Does the plan address techniques for ensuring software supportability in the development phase?

   D. Does the plan address the transition from software development to software support?

5. Did support and/or program office personnel work with configuration management to ensure the change control process (i.e. forms used, approval process, and implementation process) was suitable for the software support effort? (If so, please explain briefly.)

6. Did support and/or program office personnel work with quality assurance to ensure software supportability was included in the Quality Assurance Plan? Please explain briefly.

7. What requirements were levied on the development contractor(s) to ensure software supportability? (e.g., requirements for the support environment, software documentation, quality assurance, product evaluation, and transition)

Part III: Implementing PDSS Planning During Software Development

1. Did government support personnel attend the reviews and audits and were they able to review applicable documents prior to the actual review or audit?

2. Were government support personnel able to review or inspect code, as it was developed?

3. Were any software supportability tests performed with the developed code? (e.g., evaluating the time for selected support personnel to identify and correct a hypothetical bug)

4. Did government support personnel participate in any of the software testing to become familiar with the system? What level of participation took place? (e.g. witnessing or hands-on.)

The following questions pertain to those weapon system programs which have already fielded an operational system. If your weapon system is still under development, please skip parts IV and V.

Part IV: Transitioning from Software Development to Software Support

1. Was the PDSS transition plan followed?

2. Did the software developer provide any training?

3. Was the transition from software development to software support routine or difficult? Please explain briefly.

4. What suggestions or recommendations would you have for improving the transition process?

Part V: Post Deployment Software Support Stage

1. Is the follow on software support under a contract, performed by Air Force personnel, or a combination?

2. In what year was the software support contract awarded, or did the Air Force organization take responsibility for the software support?

3. Has supporting the software for your weapon system been routine or difficult? Please explain briefly.

4. Is there anything which could have been done differently during system development to simplify, improve, or make the system's software support more effective?

Appendix B
Guidelines for Ensuring Software Supportability in Systems Developed Under IWSM

GUIDELINES FOR ENSURING SOFTWARE SUPPORTABILITY IN SYSTEMS DEVELOPED UNDER IWSM

Table of Contents

Chapter One
Overview

1.1 Software Supportability Background

(Note: Throughout these guidelines the terms "software developer" and "developer" are assumed to mean civilian contractors or a government organization which develops software.)

In 1992, companies in the United States spent $30 billion maintaining software. This number represents anywhere from 60 to 80 percent of each company's software budget. The estimate is that, by 1995, the number will grow to 90 percent. Figure 1-1 shows past and predicted Department of Defense expenditures for embedded software. Applying the 60 to 80 percent to the 1992 figure ($29 billion), the DoD spent anywhere from $17 billion to $23 billion supporting developed or acquired software. By 1995, the estimates for DoD software support could reach $32 billion. The majority of these changes involve user enhancements to the system once it is operational. The causes of this high cost of software support are:

- Software developers are only concerned with building the new system with no regard for future supportability, while maintainers are concerned with receiving a functional system which can be easily supported.
- Once the maintenance organization has taken responsibility for the system, the maintainers must live with whatever software has been provided by the developers. If the system was developed without supportability in mind, then the maintainers will experience difficulty supporting the system.
- If the maintenance organization is separate from the development organization, then maintainers have had very little interaction with the developers and are relatively unfamiliar with the system.
- The software developer is more concerned with fulfilling contractual ob-

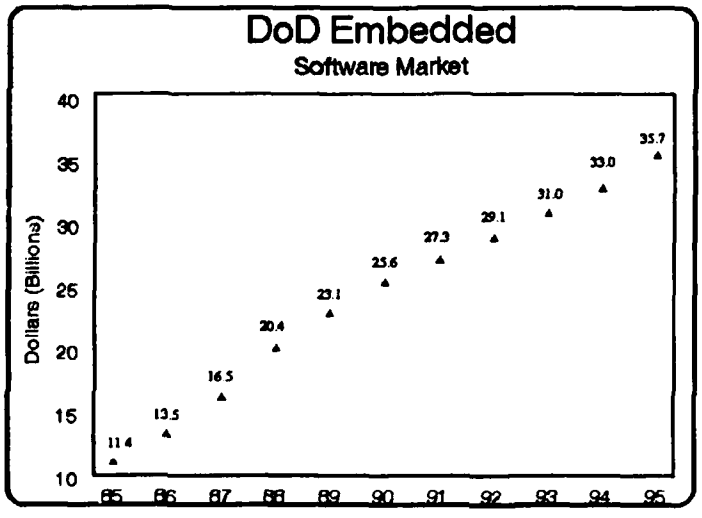


Figure 1-1

ligations (actual code and its documentation) than with providing maintainable software. Software maintainers also want a system which meets the user's performance requirements, but they (maintainers) need a system which can be easily modified in order to provide effective support.

Software maintainability is a software characteristic which reflects the degree of effort required to accomplish the following tasks:

1. Correction of errors
2. Addition of features
3. Deletion of capabilities
4. Adaptation/Modification

Maintainability is a desired characteristic of software and not just another phase of the software life cycle. While the major goal of the program office is to get the software completed on time and within budget, it should also have a goal of providing maintainable software, with particular emphasis on techniques which make enhancements to the software easier to implement. Any improvements in maintainability will reduce both the effort and cost of supporting the software.

1.2 Integrated Weapon System Management (IWSM) Background

Today, due in part to the tremendous world political changes and more austere military budgets, the process of weapon system acquisition and sustainment is evolving. IWSM has been adopted to manage this evolving process.

IWSM is defined as:
"...the AFMC management philosophy for acquiring, evolving, and sustaining our products. It empowers a single manager with authority over the widest range of decisions and resources to satisfy customer requirements throughout the life cycle of the product.".

IWSM is a management philosophy, not a process. However, even as a philosophy, it is changing the responsibility levels, management structures, partnerships, and processes of current weapon system acquisition and support organizations. Now, a weapon system will be managed by a single individual who ties together systems acquisition and sustainment, two formerly separate functions. Conceptually, it is management of a single product for each element of a system (e.g., landing gear, engines, avionics) for that system's entire life cycle.

The System Program Director (SPD) is the single manager for the weapon system throughout its life cycle. The SPD manages two groups at different ends of the life cycle: the Development System Manager (DSM) and the System Support Manager (SSM). Also, the DSMs and SSMs manage several elements of their system(s) through Integrated Product Teams (IPTs). IPTs are multidisciplinary teams empowered with the responsibility for developing and/or supporting their respective system elements. Developers and maintainers are now brought together at program start to jointly manage system software until program cancellation or decommission.

Current Post Deployment Software Support (PDSS) planning and implementation processes are impacted by this new approach. Neither the IWSM philosophy, nor AFMCP 800-60 regulate or standardize a PDSS methodology. Therefore, individual weapon system programs decide, within the IWSM conceptual framework and applicable regulations, how best to plan for and implement PDSS for their weapon systems.

1.3 Purpose

The purpose of these guidelines is to help program managers in IWSM programs ensure supportability in contractor acquired or government developed software systems. IWSM programs can benefit from the recommended techniques because there is now only one agency responsible for both development and maintenance. The SPD has an inherent interest in developing supportable software because he or she is now ultimately responsible for supporting the software. With the majority (up to 80%) of a software system life cycle's time, effort and budget going to support, the SPD should implement any techniques which would improve the maintainability of the weapon system software. The major benefits of a maintainable system are a reduced life cycle cost, and a better operational system which can be easily and effectively modified. It is cheaper to fix problems before the system becomes operational. Maintainable software has fewer problems, faster repair times, and changes to meet new requirements more efficiently.

1.4 Overview

These guidelines are divided into two sections, planning for PDSS and ensuring future software supportability. The first section describes how to build a plan which will be used during development to construct a supportable system. The second section describes techniques which can be incorporated into the plan to ensure the developed system is supportable.

Chapter Two
Planning for PDSS

2.1 Introduction

This chapter defines the Post Deployment Software Support (PDSS) plan. The chapter is divided into two sections: planning for PDSS and PDSS acquisition requirements.

The PDSS process starts during the development of a software system, where several organizations share responsibility for PDSS planning:

1. The user organization, or customer.
2. The DSM organization or System Program Office (SPO) (also the procuring agency).
3. The development organization, or developer.
4. The maintenance organization, or maintainer (also Software Support Activity (SSA)).

Together, these four groups form the weapon system acquisition and support team.

Planning is one of the most important activities to reduce the cost and risk of future software maintenance. The earlier the planning begins, the more effective the support will be. The PDSS plan outlines the procedures for maintaining the weapon system software. The plan includes:

- Organizational procedures
- Hardware resource requirements
- Software resource requirements
- Personnel requirements
- Facilities requirements

The plan also covers the program office's relationship with the user and maintainer, how requirements changes are incorporated into the system, and quality assurance and testing of each changed version.

The PDSS plan should be separate from the Computer Resources Life Cycle Management Plan (CRLCMP). The PDSS plan can be produced and maintained by the SSM, instead of being a small chapter in the much larger CRLCMP and stored in a library. The separate plan should also go into more detail than the CRLCMP on both the follow-on support and ensuring supportability in the developed software.

Assuming an SPD has been identified, whether at a Product Center or Air Logistics Center, this is the first opportunity for an IWSM organization to address future PDSS. Software development and/or maintenance personnel have the most influence in guiding the PDSS plan. This integrated approach to defining PDSS requirements forms the foundation for software supportability during development and ensures that the user and the SSA receives a product they can support.

2.2 Plan for PDSS

The PDSS plan describes how and to what extent the software will be maintained. The procedures define facilities, equipment and personnel requirements. The complete and minimum personnel requirements for follow-on support are identified, along with the full and minimum system transition personnel. SSA facility requirements along with the system hardware and software suites needs are part of the planning. If at all possible, the maintenance organi-

zation should be identified or the contract should be awarded by the time the plan is formed. If they are not identified, then the actual functions expected of the future SSA are laid out. The SSA's functions fall into the following categories:

- management and administration
- software engineering and test
- configuration management
- software generation and distribution
- technical documentation
- deployment and installation
- quality assurance

2.2.1 Management and Administration

A management structure coordinates and controls not only the SSA activities, but also the various organizations involved. An appropriate IPT structure for the particular weapon system will integrate and communicate PDSS ideas, approaches and issues. The participants come from the lowest organizational levels possible, taking responsibility for planning decisions.

Organizational PDSS responsibility agreements (possibly a Memorandum of Agreement (MOA)) are established between the user and SPD. These agreements keep everyone focused on their tasks. The agreements prevent the DSM organization from trading off software supportability requirements to preserve schedule. The SSA organization, under the SSM, is required to review all documentation, and participate in formal walkthroughs, reviews, audits, and testing during software development. The agreements also bring the user and maintainers together early through the SPD and IPT. The user and maintainer will support the software for at least 10 to 15 years after system deployment. The DSM and SSM organizations develop a synergistic relationship because they are both working towards the same goal: a high quality system.

The SPD is the hub between these parties. Communication is vital, especially when all the participants aren't collocated. IPTs are ineffective without it. The SPD must ensure that the right people from the right organizations can meet whenever needed. If face-to-face meetings are impractical, video teleconferencing, phone teleconferencing, and other media forms are good substitutes. A high quality, supportable system requires the effort of all these organizations.

2.2.2 Software Engineering and Test

This function ensures that the software maintainers will have all the necessary tools for modifying and testing the system. Even though the developer will spend a great amount of time developing a set of test data, it may not be used by the maintainers. Some reasons for this are: the test data was not a deliverable item under the development contract, the maintainers may not have the proper tools for testing software, and the data itself may be hard or impossible to understand. To ensure the test data can be reused, make sure it is a deliverable, and, if possible, have the test data jointly defined by the developers and maintainers. The plan should describe the tools needed to maintain or test the system. The following tools can be helpful:

- Cross Reference/Browsers
- Performance Analyzers
- Code Auditors
- Comparators
- Configuration Management and Version Controllers
- Requirements Tracers
- Ripple Effect Detectors

2.2.3 Software Configuration Management

Software Configuration Management (SCM) identifies how the software and its documentation will be controlled. An effective change control process improves the supportability of the software. The Software Configuration Management Plan (SCMP) is jointly produced by the developer and IPT and used throughout the life cycle. IPT membership includes SCM personnel. The SCMP should determine how the user reports problems, how the change requests are approved and who approves the requests. Then the procedures should identify how changes are scheduled, implemented and tested. Finally, the procedures identify how the new version will be released.

Configuration management also identifies which documents are required by the maintenance organization. Besides the deliverable Data Item Descriptors, the supporting organization requires data dictionaries, software designs, software requirements, specifications and documentation templates. All documentation is to be delivered in computer readable format as well as hard copy.

2.2.4 Software Generation and Distribution

This function is concerned with how SCM will distribute new software versions. This function must also address the version's accompanying documentation, and training to the user's sites. Plans and procedures must be in place to handle duplication (both magnetic and print media), packaging of the products, product delivery, and a physical audit of the packages.

2.2.5 Technical Documentation

Technical documentation requires a storage facility to maintain the volumes of documentation and magnetic media which accompany DoD software systems. The technical library must be able to contain:

- the source and object code for every release
- all test cases
- all documentation for the entire system and tools
- any tools needed to support the system.

2.2.6 Deployment and Installation

If the SSA will be responsible for installing new versions at the user's site, personnel and procedures should be identified for installation and testing of the new software, plus provisions for any other technical support or training required by the user.

2.2.7 Quality Assurance

The plan should identify whether this system is to have its own Software Quality Assurance (SQA) program or if it will be part of an organization-wide SQA program. As with the SCM plan, the SQA plan is jointly developed and used during the entire life cycle. Again, SQA personnel should be part of the IPT.
The main function of SQA is the enforcement of standards and procedures. The development and support organization may have two different standards, and the maintainers may not be willing or able to conform to a new set of standards. The supportability plan must provide for a common set of standards, including coding practices, and used throughout the life cycle. Each

organization, the user, the SPD, DSM, SSM, development and support contractor, will be responsible for enforcing the standards.

After determining what is required to support the system, these requirements are then passed on to the development contractor through the Request for Proposal.

2.3 PDSS Acquisition Requirements

This section of the supportability plan identifies the PDSS acquisition requirements, which are contractual requirements imposed upon the software development contractor which facilitate maintainability. These requirements consist of options which could reduce life cycle costs, ease the transition to maintenance, or improve maintainability or quality. When these requirements are included in the contract, life cycle costs and risks in maintenance are reduced. Many of these acquisition requirements can be taken directly from the PDSS plan. The acquisition requirements for the supportability plan are broken into:

- software environment
- technical data
- software quality
- transition requirements

2.3.1 Software Environment Requirements

Software environment requirements (which include hardware) are specified in the contract to:

- Ensure the development and support environment (including automated tools, configuration management, documentation, and network protocol) are the same, or as close as possible.
- Use existing government resources as components of the software maintenance facility.
- Provide backup systems at the software support facility.
- Limit the number of different environments and development methodologies, to include a high order language such as Ada.
- Implement a consolidated support concept, that is ensure the system can be included into a combined support facility.
- Support and encourage software reuse (design or components) where possible.

2.3.2 Technical Data Requirements

Technical data (i.e., documentation) is identified by clear and precise requirements. While development costs can be reduced by limiting the documentation requirements, not having enough of the right documentation can increase the follow on support costs. The documentation must be useful to the software maintainers, not just "eyewash". Here, communication within the IPT and joint data definition with the developer precludes data problems. The documentation must be useful to the software maintainers, not just "eyewash".

2.3.3 Software Quality Requirements

Software quality is extremely important in ensuring the maintainability of the product The concepts developed in sections 2.2.7, Quality Assurance can be passed on to the contractor The procedures must include any joint SQA ventures, such as audits or walkthroughs involving the development and support organizations. The IPT should work with the development organization to iden-

tify and establish quality requirements, a quality evaluation process, and acceptance criteria.

2.3.4 Transition Requirements

The program office, along with the development and support organization, must establish specific transition requirements. Major requirements are:

1. Ensure the SSA is staffed, trained and ready to support the system.
2. Install and test the hardware.
3. Install and test the software suite, including tools and databases.
4. Ensure all Configuration Management functions including updated documentation and CM records are in place.
5. Secure any licensing agreements or warranties.
6. Ensure the SSA is ready to support the user with no interruption in service.

Once PDSS planning and acquisition requirements are defined, then the focus shifts to ensuring supportability during development.

2.4 Planning Summary

Table 2-1 on the following page summarizes PDSS planning, SSA functions, and acquisition requirements.

Table 2-1: PDSS Planning Activities

PDSS Planning	
Section	**Contents**
Personnel	Full and minimum staffing for both transition and follow-on support.
Facilities	Facilities required to house all equipment and personnel.
Hardware and Software Suites	All hardware and software required to support the system, plus any backups.
SSA Functions	
Section	**Contents**
Management and Administration	Management and IPT structure; agreements; personnel, software, hardware, and facility requirements.
Software Engineering and Test	Any tools needed to support, test, and analyze the software.
Configuration Management	Deliverable documents for support; define change control process.
Software Generation and Distribution	Delivery and training for new software versions.
Technical Documentation	Define repository for documents, code, and tools
Deployment and Installation	Personnel and procedures needed for software installation and test.
Quality Assurance	Establish a common set of standards and policies.
Acquisition Requirements	
Section	**Contents**
Software Environment	Common development and support environments; provide backup systems; limit the number of development methodologies; support reuse
Technical Data	Documents needed for support listed as deliverables.
Software Quality	Establish quality requirement, evaluation process, and acceptance criteria
Transition	Installation, training, documentation, configuration management records, and warranties.

Chapter Three
Ensuring Supportability During Development

3.1 Introduction

This chapter provides techniques for ensuring software supportability during the development cycle. Software quality and supportability go hand-in-hand during development. Any methods to improve quality improves supportability and any attempts to improve supportability result in a higher quality product. Software supportability and quality cannot be ensured by contract specifications. Software maintainability and quality can be improved by using these techniques:

- high order languages
- reusable and commercial off-the-shelf software
- modularity
- reviews
- standards
- testing

3.2 High Order Languages

Using a high order language improves software quality and supportability. The Air Force mandated the use of the programming language Ada in 1990. The language has a number of built in facilities which simplify the transition from design from implementation, thereby reducing the intellectual effort and the errors which accompany this effort. Ada is designed to make use of the principles of information hiding and data abstractions, which can improve quality and maintainability. Ada programs interact with each other through the package specifications, while hiding the implementation details from other programs in the package body. The interfaces between the programs must be well defined and thought out, which help to ensure a higher quality product. With data abstraction, there is a single definition of a data object coupled with all the operations that can be performed on the object. This abstraction simplifies any changes made to the data object.

3.3 Reusable and Commercial Off-The-Shelf Software

Reusable and commercial off-the-shelf (COTS) software inherently increases supportability and quality because each has been tested and implemented in other domains. COTS requires little or no modification, while reusable components contain fewer errors which simplifies testing and reduces the amount of modification of the software. However, if the COTS needs to be modified for any reason, then problems could arise with the effort and expense of the changes. The issue of data rights for COTS can also complicate matters. The IPT should identify whether COTS and reusable components will be part of the PDSS scheme.

3.4 Modularity

Another maintainability technique used during development is modularity; programs should be composed of small units which have a high degree of cohesion and low degree of coupling. Cohesion is the degree to which functions within the module are bound together. Coupling is degree to which modules are dependent on each other.

The Osborne article referenced provides more details on cohesion and coupling.

3.5 Reviews and Audits

Reviews are conducted so the DSM and the user can monitor the developer's progress. These reviews and audits assure a uniformity across the software, which is critical when someone other than the original programmer must maintain the code. The review determines if the product is meeting specific performance, design, and verification requirements and that the entire task can still be completed within the projected schedule and allocated budget. Audits are more closely related to inspections where SQA inspects code modules or test procedures for compliance with standards. To ensure the system is supportable, and to become familiar with the system, maintenance personnel must carefully read all documentation and participate in each review. The documentation and accompanying reviews are intended for technical people, not managers. Although SQA evaluates the documents for standardization, the documents must be passed on to the appropriate personnel, i.e., programmers and testers, who evaluate the documents against the following criteria:

- Internal consistency.
- Understandability.
- Traceability to the indicated documents.
- Consistency with the indicated documents.
- Appropriate analysis, design, or coding techniques used.
- Appropriate allocation of sizing and timing resources.
- Adequate test coverage of requirements.

Although support managers should attend the reviews to establish good working relationships with their corresponding development managers, it is the technical personnel who make the greatest contribution to the quality of the system at a review.

3.6 Standards

Standards are essential for establishing a common maintenance environment. This environment provides a common ground for reviewing another programmer's work, understanding another's code, and ultimately changing another person's program. The standards are divided into:

- development standards.
- coding practices.
- comments.
- maintainability checklists.

3.6.1 Development Standards

In the Air Force, the software development standard most often used for weapon systems is DOD-STD-2167A, Military Standard for Defense System Software Development. Use of the standard is meant to ensure that all contractors or organizations use the same uniform requirements in developing software, along with the means for establishing, evaluating, and maintaining quality in the software and its associated documents. The military is in the process of replacing DOD-STD-2167A with MIL-STD-498, Software Design and Documentation.

3.6.2 Coding Practices

Good coding practices improve maintainability by providing a structure and framework in which systems can be developed and maintained in a clearer manner. Management or software quality assurance places the coding practices requirement on programmers. An example list of practices may include:

- Language requirements - Use Ada. The language is mandatory unless

some other language can be shown to be more cost-effective.

- Code complexity - Use structured programming while avoiding undesirable language forms (programmer tricks which someone else might not understand).
- Modularity - Break a large procedure into smaller ones.
- Naming conventions - Use names which reflect the content of the variable or the structure of the program or data or both.
- Comments - Use standard comments for each subprogram.
- Traceability - Trace each section of the code back to the specifications.

The Department of Defense uses MIL-STD-1815A, Military Standard for the Ada Programming Language. The standard provides specifications for:

- Declarations and types.
- Names and expressions.
- Statements.
- Subprograms.
- Packages.
- Tasks
- Program structure.
- Exceptions.
- Generic Units.
- Input and output.

The Ada Quality and Style Guide (QSG) specifies the form and meaning of Ada programs to facilitate supportability, portability and reusability. When used effectively, the QSG a common ground for both the developers and maintainers to work.

3.6.3 Maintainability Checklists

A maintainability checklist provides an actual tool to evaluate the code from a maintainability perspective. Software designed against this checklist shows the following characteristics:

1. Consistency - correlate and contain uniform notation, terminology, and symbology while meeting the design objectives and requirements.
2. Testability - establish verification criteria and support evaluation of its performance, preferably at the module level.
3. Modularity - partition the software into reasonably sized parts, components, and modules; blocks of code are completely independent of all other modules within the program.
4. Descriptiveness - include information regarding the software's objectives, assumptions, inputs, processing, outputs, revision status, etc. The software avoids ambiguity and is readily understood. This includes code commenting and header information.
5. Changeability - Must be able to physically change the information, computational functions, data storage, or execution time can be readily accomplished.

3.6.4 Comments

Good code comments also improve the maintainability of the system by providing readability and valuable information on the module, including its design. This is important when someone other than the developer must modify the code. Comments should include the following:

- Why the code was implemented this way.
- How this module affects other modules.
- Any potential problems.
- When the changes were made.
- Who made the changes.
- What the changes were.

3.7 Code Walkthroughs and Inspections

A software inspection is a review of a programmer's or tester's work to find problems and improve quality by allowing the workers to recognize and fix their own problems before the system becomes operational. An inspection or walkthrough by peers can catch problems which the workers could not find, and which may not be discovered in testing. The inspection also ensures that both parties technically agree on the work and it meets predefined criteria. It may not be accurate to call the process between development and maintenance personnel an inspection, since the term *inspection* could put the developers on the defensive. An appropriate term is formal walkthrough or just walkthrough.

The walkthrough follows these principles:

1. The walkthrough is a formal process with a system of checklists and defined roles. The roles include moderator, producer, reviewers, and recorder.
2. Checklists and standards are developed and tailored for each participant.
3. The participants prepare for the walkthrough in advance.
4. The purpose of the walkthrough is to discover problems.
5. The walkthroughs are for technical people, not their managers.
6. The results are documented.

Because the walkthroughs involve personnel from three different organizations (developer, maintainer, and DSM), the moderator must be from the DSM, who ensures that both organizations are treated fairly and that the meeting doesn't turn into a finger pointing or shouting match.

Scheduling formal walkthroughs of the developed products with corresponding maintenance personnel improves maintainability in three ways:

1. The maintainers are given the product to study before the walkthrough, which provides excellent training by familiarizing the maintainers with the code.
2. The maintainers provide an independent opinion, because they weren't the original authors.
3. The maintainers have a very high stake in the outcome: the better the walkthrough, the easier the maintenance.

The results of the walkthrough should be documented, and the recommendations should be implemented after the walkthrough. A follow up walkthrough is advisable to see if the recommendations were in fact implemented.

When coding ends, the focus shifts to supportability techniques in testing.

3.8 Testing

A system's maintainability is increased by thorough testing of the system. Thorough testing is accomplished through enhancing the test plans and procedures. A rigorous review and inspection process by the IPT and maintainer enhances these plans and procedures. Test plans must account for all requirements and conditions. Procedures must attempt to increase the numbers of errors discovered and corrected, reduce the likelihood of future errors, and ensure a higher quality system. The testers in the maintenance organization must review and inspect the test plans, plus participate in the development of the test cases. The testers must also attend the Preliminary Design Review, the Critical Design Review, and the Test Readiness Review to evaluate

the plans and procedures. By participating in the actual testing, the maintainers gain training on the system, a better understanding of how the system works, and how it is tested. The test procedures are passed on to the support organization for reuse or future regression testing.

During the error correction phase, the IPT continually evaluates the system's configuration management change process is continually evaluated to ensure changes are properly identified, validated, and implemented. The software correction turnaround time should also be tracked so errors are identified and corrected in a reasonable amount of time.

3.9 Supportability Techniques in the Development Cycle

With software supportability, timing is just as important as method. The development cycle is broken into five phases: system requirements and design, software requirements analysis, preliminary design, detailed design, CSCI test, and system integration and test. Table 3-1 on the following page shows what the maintainers can do in each phase of the development cycle.

3.10 Summary

These guidelines have presented a plan for improving supportability. The plan stresses planning for personnel and facilities, the actual functions that will be performed by the Software Support Activity (SSA), and supportability requirements that can be passed on to the contractor. The guidelines also include supportability techniques that can be used during development. By using the information and techniques presented here Software Support Managers can improve the both the cost and effort of maintaining their software systems.

Table 3-1 Supportability Techniques in the Development Cycle

Phase	Software Support Personnel			
	SSM	Programmers	Testers	QA
System Requirements and Design	Write PDSS Plan, Attend System Requirements Review (SRS)	Read System/Segment Specification, Attend SRS		Read SSS
Software Requirements Analysis, Preliminary Design	Conduct Design Walkthrough, Attend Preliminary Design Review (PDR)	Read Preliminary Software Design Document (SDD), Attend PDR, Participate in Design Walkthroughs	Develop Software Test Plan (STP), attend PDR	Read Preliminary SDD, STP
Detailed Design	Conduct design and test case walkthrough, attend Critical Design Review (CDR)	Read detailed SDD, Attend CDR, Participate in design walkthroughs	Develop Software Test Procedures, Construct test cases with developer, Attend CDR, Participate in test case walkthrough	Read Software Test Procedures, SDD
Code, CSU Test, CSC Integration Test	Conduct code walkthrough	Participate in code walkthroughs, evaluate code against maintainability checklist	Participate in testing	Joint audits with developer QA to enforce standards & practices, Evaluate CM during error correction
CSCI Test, System Integration Test		Observe error correction by development programmers	Participate in Testing	Evaluate CM during error correction

References

1. Arnold, Robert S. "Improving Software Acquisition to Facilitate Software Maintenance", in 1987 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1987.

2. Department of the Air Force. IWSM Guidebook. AFMCP 800-60. Dayton OH: HQ AFMC, 1993.

3. Department of Defense. Guidelines for the Successful Acquisition of Computer Dominated Systems and Major Software Developments. Washington DC. GPO, 1992.

4. Department of Defense. Military Standard for Defense System Software Development. DOD-STD-2167A. Washington DC: GPO, 1988.

5. Department of Defense. Military Standard for the Ada Programming Language. MIL-STD-1518A. Washington DC: GPO, 1983.

6. Department of Defense. Mission Critical Computer Resources Software Support. MIL-HDBK-347. Washington DC: GPO, 1990.

7. Department of Defense. Mission Critical Computer Resources Management Guide. Washington DC: GPO, 1990.

8. Edelstein, D. Vera and Salvatore Momone. "A Standard for Software Maintenance", Computer, 37:82 - 83 (June 1992).

9. Glass, Robert L., Building Quality Software. Englewood Cliffs NJ: Prentice-Hall, 1992.

10. Hager, James A., "Developing Maintainable Systems: A Full Life-Cycle Approach", in 1989 IEEE Conference on Reliability and Maintainability. New York: IEEE Computer Society Press, 1989.

11. Humphrey, Watts S. Managing the Software Process. Reading MA: Addison-Wesley Publishing, 1990.

12. Lawlis, Lieutenant Colonel Patricia K. Air Force Institute of Technology, Wright Patterson Air Force Base OH. Personal Correspondence. 29 July 1993.

13. Osborne, Wilma M. "Building and Sustaining Software Maintainability", in 1987 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1987.

14. Sherer, Susan A. "Cost Benefit Analysis and the Art of Software Maintenance", in 1992 IEEE Conference on Software Maintenance. Los Alamitos CA: IEEE Computer Society Press, 1992.

15. Sunday, David A. "Software Maintainability - A New 'ility", in 1989 IEEE Conference on Reliability and Maintainability. New York: IEEE Computer Society Press, 1989.

16. Vollman, Thomas. "Transitioning From Development to Maintenance", in 1990 IEEE Conference on Reliability and Maintainability. Los Alamitos CA: IEEE Computer Society Press, 1990.

Bibliography

1. "A Checklist of Major Aeronautical Systems", Air Force Magazine, 76: 56-57 (January 1993).

2. "A Checklist of Major Electronic Systems", Air Force Magazine, 75: 34-35 (July 1992).

3. "A Checklist of Space Systems", Air Force Magazine, 75: 34 (August 1992).

4. Abbott, Stephen, F. "Standardized Code", in 1983 IEEE Workshop on Software Maintenance. Silver Spring MD: IEEE Computer Society Press, 1983.

5. Arnold, Robert S. "Improving Software Acquisition to Facilitate Software Maintenance", in 1987 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1987.

6. Department of Defense. Guidelines for the Successful Acquisition of Computer Dominated Systems and Major Software Developments. Washington DC: GPO, 1992.

7. Department of Defense. Military Standard for Defense System Software Development. DOD-STD-2167A. Washington DC: GPO, 1988.

8. Department of Defense. Military Standard for the Ada Programming Language. MIL-STD-1518A. Washington DC: GPO, 1983.

9. Department of Defense. Mission Critical Computer Resources Software Support. MIL-HDBK-347. Washington DC: GPO, 1990.

10. Department of Defense. Mission Critical Computer Resources Management Guide. Washington DC: GPO, 1990.

11. Department of the Air Force. IWSM Guidebook. AFMCP 800-60. Dayton OH: HQ AFMC, 1993.

12. Edelstein, D. Vera and Salvatore Momone. "A Standard for Software Maintenance", Computer, 37:82 - 83 (June 1992).

13. Emory, C. William and Donald R. Cooper. Business Research Methods (Fourth Edition). Homewood IL: Richard D Iwrin, Inc., 1991.

14. Fritz, Rodger L. and Fred Shocket. "LAMPS - Demonstrated Maintainability Through Application of MIL-SPEC Software Development Techniques", in 1988 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1988.

15. Glass, Robert L. Building Quality Software. Englewood Cliffs NJ: Prentice-Hall, 1992.

16. Hager, James A. "Developing Maintainable Systems: A Full Life-Cycle Approach", in 1989 IEEE Conference on Reliability and Maintainability. New York: IEEE Computer Society Press, 1989.

17. Humphrey, Watts S. Managing the Software Process. Reading MA: Addison-Wesley Publishing, 1990.

18. Nicholas, John M. Managing Business and Engineering Projects. Englewood Cliffs NJ: Prentice-Hall, Inc., 1990.

19. Osborne, Wilma M. "Building and Sustaining Software Maintainability", in 1987 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1987.

20. Pigoski, Thomas M. and Craig A. Cowden. "Software Transition: Experience and Lessons Learned", in 1992 IEEE Conference on Software Maintenance. Los Alamitos CA: IEEE Computer Society Press, 1992.

21. Sherer, Susan A. "Cost Benefit Analysis and the Art of Software Maintenance", in 1992 IEEE Conference on Software Maintenance. Los Alamitos CA: IEEE Computer Society Press, 1992.

22. Sunday, David A. "Software Maintainability - A New 'ility", in 1989 IEEE Conference on Reliability and Maintainability. New York: IEEE Computer Society Press, 1989.

23. "USAF Human Systems Checklist", Air Force Magazine, 75: 56-57 (December 1992).

24. Vollman, Thomas "Transitioning From Development to Maintenance", in 1990 IEEE Conference on Reliability and Maintainability. Los Alamitos CA: IEEE Computer Society Press, 1990.

25. Wu, Chung-Fern. "Information System Development Audits and Software Maintenance", in 1987 IEEE Conference on Software Maintenance. Washington DC: IEEE Computer Society Press, 1987.

Vita

(Forrest F. Butts III)

Captain Forrest F. Butts III was born on 30 January 1962 in Montgomery, Alabama. He graduated from Prattville High School in 1980. He attended the University of Alabama in Tuscaloosa, Alabama earning a Bachelor of Science degree in Computer Science in December 1984. He then attended Officer Training School at Lackland AFB, Texas receiving his commission on 13 September 1985. His first duty assignment was to the 1912th Computer Systems Group at Langley AFB, Virginia. There he served as a World Wide Military Command and Control System Database Programmer, Joint System Test Team Chief for the Tactical Air Control System (TACS) Executive Officer, and Chief of Software Quality Assurance for the Contingency TACS Automated Planning System. Capt Butts' next duty assignment was with the Communication Systems Center's Operating Location "A", where he served as Chief of Communications Software, responsible for the development and maintenance of the Communications Front End Processor,which transmits and receives all weather data for Air Force Global Weather Central. His next position was as Chief of Operating Location "A", where he was responsible for the development and maintenance of communications software on three mainframe systems. Capt Butts entered the School of Systems and Logistics in May of 1992 in pursuit of a Master of Science in Software Systems Management.

Permanent Address:
1304 Huie St.
Prattville, Al 36066

Vita

(Anthony C. Johndro)

Captain Anthony C. Johndro was born on 5 September 1963 in Grand Falls, New Brunswick, Canada. He graduated from Limestone High School in Limestone, Maine in 1981 and attended Boston University, graduating with a Bachelor of Science in aerospace engineering in May 1985. He then attended the USAF Officer Training School, received a commission in the USAF and served his first tour of duty at Los Angeles AFB. He began his career as a Space Defense Mission Analysis Officer for the Space Division (AFSC) Office of Plans and Advanced Programs where he analyzed operational needs for space control, space surveillance, and space battle management until March 1989. He was then assigned to the Over-the-Horizon Backscatter Radar System Program Office, Hanscom AFB, as a Radar Subsystem Manager responsible for the development, unit-level, and system-level testing of several radar software subsystem components. He was then chosen to serve as Chief, Software Maintenance, to define, specify, and manage interim contractor support for the planning, scheduling, budgeting, development, and release of two major software version releases for the radar system. His responsibilities included planning, scheduling, and budgeting the transfer of the radar system's Software Maintenance Facility to the supporting Air Logistics Center. He was then selected to serve as Chief, Computer Resources, where he assumed greater responsibilities for the entire radar computer hardware and software development, interim support and support planning. His duties included chairmanship of the Computer Resources Working Group to plan the transition and future support of the radar's software until entering the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May 1992.

Permanent Address:
RD 5, Box 313
Lewistown, PA 17044

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE
GUIDELINES FOR ENSURING SOFTWARE SUPPORTABILITY IN SYSTEMS DEVELOPED UNDER THE INTEGRATED WEAPON SYSTEM MANAGEMENT CONCEPT

5. FUNDING NUMBERS

6. AUTHOR(S)
Forrest F. Butts, III, Capt, USAF
Anthony C. Johndro, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/LAS/93D-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ASAF/Acquisition
SAF/AQKS, 5D544
1060 Air Force Pentagon
Washington, DC 20330-1060

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This thesis studied the software maintenance planning and practices of the pilot Integrated Weapon System Management (IWSM) programs. Before IWSM, a System Program Office (SPO) acquired an Air Force weapon system then passed it to an Air Logistics Center (ALC) for follow-on support. The ALC was forced to maintain the software despite its questionable maintainability. The SPO de-emphasized maintainability because maintenance was an ALC responsibility and building maintainable software increased development costs and lengthened schedules. The IWSM philosophy closes the gap between development and maintenance. A System Program Director (SPD), who oversees both system development and maintenance, has an inherent interest in developing maintainable software because he or she is now also responsible for supporting it. This research was accomplished through a literature review of current maintainability plans and practices, followed by a survey of the pilot IWSM programs. This information was combined to form draft guidelines for ensuring software maintainability. The draft guidelines were then validated by experts in the field of software maintenance who offered opinions and recommendations on the guidelines. The guidelines stress both up front planning and techniques for improving maintainability during software development. The final guidelines are presented.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Systems Management, Computer Programs, Maintenance Management, Logistics Planning, Maintainability, Teams (Personnel)	94
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GSS/LAS/93D-1

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: DEPARTMENT OF THE AIR FORCE, AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT PATTERSON AFB OH 45433-7765

1. Did this research contribute to a current research project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____

Name and Grade Organization

_____ _____

Position or Title Address